

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta informačních technologií

ITU

Vývoj aplikace vo frameworku CodeIgniter

Obsah

1	Úvod.....	3
2	Základné informácie o frameworku	3
3	Prevádzkové prostredie	3
4	Vytvorenie aplikácie prostredníctvom frameworku CodeIgniter	3
4.1	Architektúra a príprava vývojového prostredia	3
4.2	Vytvorenie dátovej vrstvy.....	4
4.3	Vytvorenie aplikačnej vrstvy	4
4.3.1	Konfigurácia a spozajzdnenie frameworku	4
4.3.2	Implementácia aplikácie prostredníctvom frameworku	5
4.3.3	Hlavička a päta frontendu	5
4.3.4	Informatívne podstránky	6
4.3.5	Vytvorenie kontroleru k informatívnym podstránkam	6
4.3.6	Dynamická časť aplikácie	7
4.3.7	Zobrazenie kultúrnych udalostí	7
4.3.8	Odstránenie kultúrnych udalosti	9
4.3.9	Pridanie kultúrnych udalosti.....	10
5	Záver.....	13
6	Bibliografia.....	13

1 Úvod

V posledných rokoch sa čoraz populárnejšími stávajú webové aplikácie, ktoré nadobudli obrovský progres a to aj vďaka vývoju skriptovacieho jazyka PHP. Vďaka tomu že skriptovací jazyk PHP ponúka vývojárovi oveľa väčšie možnosti ako bežné skriptovacie jazyky, sa tento jazyk stal jednou z najpoužívanejších technológií pre vývoj dynamických webových aplikácií. Vyvíjanie takejto aplikácie môže byť v samotnom jazyku PHP náročné i zdĺhavé, a práve preto existuje niekoľko frameworkov, ktorých úlohou je najmä prácu vývojára uľahčiť. My si v tomto článku predvedieme jeden z najpopulárnejších frameworkov, ktorý tieto spomenuté ciele spĺňa a jeho názov je Codeigniter. Codeigniter je v rámci ostatných, podobných frameworkov populárny najmä kvôli jeho jednoduchosti a rýchlosti. My si najprv k tomuto frameworku povieme niekoľko základných skutočností a následne si ukážeme ako je pomocou neho možné vytvoriť jednoduchú dynamickú webovú aplikáciu.

2 Základné informácie o frameworku

Framowork Codeigniter (ďalej aj ako „CI“) je voľne dostupný a určený pre vývoj dynamických webových aplikácií v skriptovacom jazyku PHP. Je založený na architektonickom princípe MVC (Model-View-Cotroller) a bol vyvíjaný najmä spoločnosťou EllisLab, pričom v roku 2014 tento vývoj prebrala spoločnosť British Columbia Institute of Technology. Ako je už aj vyššie spomenuté, CI ponúka sady nástrojov a knižnice s jednoduchým rozhraním, pre úsporu času vývojára. Najnovšou hotovou verziou frameworku je Codeigniter 3.1.11, pričom verzia Codeigniter 4.0.0 je práve vo vývoji. (1)

3 Prevádzkové prostredie

CI je aplikácia napísaná v jazyku PHP. Zdrojový kód CI 3.1.11 je založený na PHP a jeho verzii 5.6. Pre prevádzkovanie aplikácie je potrebný webový server, na ktorom je možné spracovať PHP skripty. Takýmto serverom je napríklad Apache.

Pre rozbehnutie frameworku je potrebné skopírovať hlavný priečinok CI so všetkými zdrojovými súborami na adresára odkiaľ číta webový server. Najdôležitejším podpriečinkom je zložka application, kde budú uložené aj zdrojové súbory vývojára, a kde sa takisto nachádzajú podpriečinkom config, ktoré obsahujú najdôležitejšie súbory pre konfiguráciu aplikácie.

Najdôležitejšie z týchto konfiguračných súborov sú:

- config.php – nastavenia rôzneho druhu správneho chodu aplikácie
- autoloader.php – pomocou neho sa automaticky nahrávajú knižnice, modely atď.
- database.php – nastavenia pripojenia k databáze
- routes.php – pomocou neho je možné prepisovať URI dotazy

Bližšie informácie k týmto konfiguračným súborom a ich použitie si ukážeme v nasledujúcej kapitole

4 Vytvorenie aplikácie prostredníctvom frameworku Codeigniter

V tejto kapitole si ukážeme, ako na základe spomínaného frameworku vytvoriť jednoduchú aplikáciu. Naším hlavným cieľom bude vytvoriť podstránku webovej aplikácie pre spoločnosť, ktorá organizuje rôzne divácke udalosti, ako sú premietanie filmov, divadelné predstavenia, či rôzne koncerty. Na našej podstránke bude možné spravovať tieto udalosti resp. diela a to pridávaním, odoberaním a prezeraním týchto diel. Aby sme tieto diela mohli uchovávať vytvoríme si aj jednoduchú databázu na uchovávanie týchto informácií. V tomto článku sú iba ukážky úryvkov zdrojového kódu, preto v prípade potreby alebo rôznych problémov sú zdrojové súbory nami vytváranej aplikácie dostupné na tomto odkaze:

https://github.com/jozef507/itu_CI_app.

4.1 Architektúra a príprava vývojového prostredia

Našu jednoduchú aplikáciu vytvoríme na základe trojvrstvovej architektúry (three-tier architecture), ktorú si pre naše potreby budeme simulovať prostredníctvom softvéru XAMPP, pomocou ktorého sme schopní vytvoriť jednoduchý lokálny webový server Apache a takisto MySQL/MariaDB databázu. Tento softvér je voľne dostupný pre platformy operačných systémov Windows, Linux a OS X na tomto odkaze:

<https://www.apachefriends.org/index.html>.

Po nainštalovaní tohto softvéru stiahneme aj súbory frameworku CI, verzie 3.1.11, na tomto odkaze: <https://codeigniter.com/en/download>. Po stiahnutí v prípade archivovaného súboru súbory rozbalíme, potom pre jednoduchosť premenujeme koreňový adresár na „ci“ a tento adresár presunieme do zložky htdocs, ktorý patrí novo-nainštalovanému softvéru XAMPP. Následne otvoríme už presunutý adresár ci v ľubovoľnom vývojovom prostredí vhodnom pre vývoj webových aplikácií. Odporúčame použiť voľne dostupné vývojové prostredie SublimeText (dostupný na odkaze: <https://www.sublimetext.com/3>). Skôr ako začneme je potrebné v softvéri XAMPP spustiť Apache Web Server a MySQL Database.

4.2 Vytvorenie dátovej vrstvy

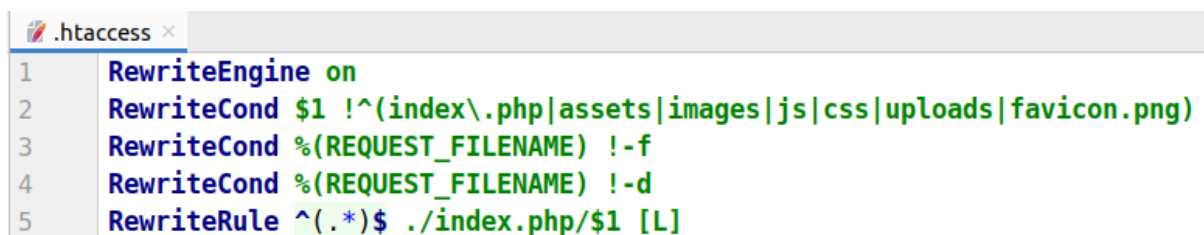
Ešte pred tým než začneme s vývojom aplikačnej vrstvy vytvoríme si MySQL databázu v systéme phpMyAdmin, ktorý nám spustený a „našartovaný“ softvér XAMPP ponuka. Dostaneme sa k nemu načítaním odkazu „localhost/phpmyadmin“ v ľubovoľnom internetovom prehliadači. Následne klikneme na tlačidlo „SQL“ v hlavnej hornej lište a zobrazí sa nám vstup pre vloženie skriptu v jazyku SQL. Vložíme do neho zdrojový kód nášho skriptu, dostupný aj na odkaze portálu GitHub, uvedenom v úvode tejto kapitoly v súbore itu_app_db.sql. Následne klikneme na tlačidlo „Vykonaj“ a tým sa nám vytvorí databáza s názvom „apka“, s tabuľkou pre uchovanie kultúrnych diel s niekoľkými vloženými údajmi (dielami).

4.3 Vytvorenie aplikačnej vrstvy

Konečne môžeme prejsť k vývoji hlavnej časti našej aplikácie. Ako prvé si ukážeme ako si spojzdníť framework CI jeho vhodnou konfiguráciou a následne prejdeme k samotnému vývoju aplikácie.

4.3.1 Konfigurácia a spojzdnenie frameworku

Ako prvé vám vysoko odporúčame, vytvoriť si v koreňovom adresári našej aplikácie (resp. frameworku) ci vytvoriť súbor „.htaccess“ so zdrojovým kódom, ktorý môžete vidieť na nasledujúcom obrázku (tento súbor nájdete aj v priložených zdrojových súboroch na odkaze portálu GitHub uvedenom vyššie). Tento súbor nám v aplikácii zabezpečí správne načítanie jednotlivých podstránok našej aplikácie prostredníctvom URI odkazov.



```
1 RewriteEngine on
2 RewriteCond $1 !^(index\.php|assets|images|js|css|uploads|favicon.png)
3 RewriteCond %(REQUEST_FILENAME) !-f
4 RewriteCond %(REQUEST_FILENAME) !-d
5 RewriteRule ^(.*)$ ./index.php/$1 [L]
```

Ako ďalšie prevedieme hlavnú konfiguráciu pomocou konfiguračných súborov frameworku v podadresári ci/application/config.

Najprv prejdeme k súboru config.php. V ňom nastavíme prvky ‚base_url‘ a ‚index_page‘ podľa \$config tak, ako to je uvedené na ďalšom obrázku. Tieto nastavenia nám umožnia pristupovať k našej aplikácii z prehliadača prostredníctvom URL odkazu ‚http://localhost/ci‘.

```
$config['base_url'] = 'http://localhost/ci';
$config['index_page'] = '';
```

Ďalej môžeme prejsť ku konfiguračnému súboru autoload.php. V ňom použijeme nastavenie konfiguračných prvkov ‚libraries‘ a ‚helper‘ ukázane na nasledujúcej ukážke zdrojového kódu. Nastavením týchto prvkov sa nám v aplikácii automaticky načítajú knižnice a pomocné funkcie ktoré pri vývoji aplikácii neskôr použijeme.

```
$autoload['libraries'] = array('form_validation');
$autoload['helper'] = array('url', 'form');
```

Ako k poslednému z týchto konfiguračných súborov prejdeme k súboru database.php. Tento súbor sa stará o nastavenie pripojenia k databáze, s ktorou má aplikácia vyvíjaná v danom frameworku pracovať. Najdôležitejšou časťou je nastavenie prvku ‚default‘, ktorý predstavuje pole. Jeho nastavenie môžeme vidieť nižšie. Ako ‚hostname‘ resp. cestu používame ‚localhost‘, ako prihlasovacie meno užívateľa, ktorým

sa na databázu pripájame používame ‚root‘. Meno databázy na ktorú sa chceme pripojiť nastavujeme meno databázy, ktorú sme pre naše potreby už vytvorili.

```
$db['default'] = array(
    'dsn' => '',
    'hostname' => 'localhost',
    'username' => 'root',
    'password' => '',
    'database' => 'apka',
    'dbdriver' => 'mysqli',
    'dbprefix' => '',
    'pconnect' => FALSE,
    'db_debug' => (ENVIRONMENT !== 'production'),
    'cache_on' => FALSE,
    'cachedir' => '',
    'char_set' => 'utf8',
    'dbcollat' => 'utf8_general_ci',
    'swap_pre' => '',
    'encrypt' => FALSE,
    'compress' => FALSE,
    'stricton' => FALSE,
    'failover' => array(),
    'save_queries' => TRUE
);
```

4.3.2 Implementácia aplikácie prostredníctvom frameworku

Ako sme si už v úvode tohto článku povedali, CI je založený na princípe MVC (Model-View-Controller). Preto jediné podadresáre adresára ci/application, ktoré nás budú pri implementácii zaujímať sú adresáre controllers, models a views. Do týchto priečinkov budeme ukladať nami vyvíjané zdrojové súbory. Na začiatok si z týchto priečinkov odstránime frameworkom predložené súbory s názvami index a welcome. To urobíme hlavne preto, aby nás ako začiatčníkov ich prítomnosť zbytočne nemiatla. Čo sa týka frameworku tak zdrojové súbory kontrolerov sú povinné, zatiaľ čo zdrojové súbory modelov a vzhľadov (views) sú voliteľné. Úlohou kontrolerov je riadenie procesov aplikácie (backendu), ktoré sa vykonávajú na pozadí a takisto povolávajú vzhľady (views), ktoré sa užívateľovi majú zobrazit'. Čo sa týka zdrojových súborov modelov, budeme ich používať pre implementáciu funkcií pracujúcich s databázou. V zdrojových súboroch vzhľadov (views) budeme naopak implementovať vzhľad užívateľského rozhrania aplikácie resp. frontend.

4.3.3 Hlavička a päta frontendu

Z dôvodu aby sme nemuseli pri každej podstránke vytvárať/implementovať tú istú hlavičku a päť, si tieto elementy vzhľadu našej aplikácie vytvoríme v samostatných súboroch. Tieto súbory, ako už správne tušíte, vytvoríme v zložke views a pomenujeme ich ako header.php a footer.php. Súbory implementujeme v jazyku HTML, doplnené o niekoľko štýlových úprav jazyka CSS.

Keďže naším cieľom je ukázať ako sa vytvára dynamická webová aplikácia, nebudeme so štylistikov HTML elementov strácať čas, a pre túto štylistiku použijeme CSS framework nazývaný Bootstrap a v rámci neho jeho jednu Bootswatch tému (dostupnú na odkaze: <https://bootswatch.com/flatly/>). Túto tému uložíme v novo-vytvorenej zložke css ktorú sme umiestnili do koreňového adresára aplikácie. Načítanie tohto CSS súboru prevedieme v header.php, pomocou vstavanej funkcie ‚base_url‘, ktorá poukazuje na predvolenú URL adresu, ktorú sme nastavili v konfiguračnom súbore config.php.

```
<head>
    <title>Diela</title>
    <link href="<?php echo base_url(); ?>css/bootstrap.min.css" rel="stylesheet">
</head>
```

Okrem toho bude súbor header.php obsahovať aj hlavnú navigačnú lištu našej aplikácie. Ta bude obsahovať ponuku jednotlivých podstránok. Okrem podstránky ‚Diela‘ pre spravovanie kultúrnych diel,

pridáme odkazy aj na podstránky ‚Domov‘ a ‚O nás‘, ktoré budú statické a ich účel bude čisto informatívny. Môžeme vidieť, že pre štylizáciu tejto navigačnej lišty využívame triedy frameworku bootstrap importovanom vyššie.

```
<ul class="navbar-nav mr-auto">
    <li class="nav-item">
        <a class="nav-link" href="php echo base_url(); ?">Domov</a>
    </li>
    <li class="nav-item">
        <a class="nav-link" href="php echo base_url(); ?&gt;shows/index"&gt;Diela&lt;/a&gt;
    &lt;/li&gt;
    &lt;li class="nav-item"&gt;
        &lt;a class="nav-link" href="<?php echo base_url(); ?&gt;pages/about"&gt;O nás&lt;/a&gt;
    &lt;/li&gt;
&lt;/ul&gt;</pre
```

Súbor header.php bude už inak obsahovať iba html element, ktorý bude obaľovať obsah jednotlivých podstránok. Čo sa týka súboru footer.php jeho obsahom bude iba ukončenie html elementov, ktoré boli načaté a neukončené v header.php.

4.3.4 Informatívne podstránky

Vzhľad podstránok ‚Domov‘ a ‚O nás‘ implementujeme v súboroch home.php resp. about.php, takisto v zložke views. Vytvárame ich preto aby naša webova aplikácia nebola príliš „holá“, ale najmä preto, aby sme si na nich demonštrovali prístup k niekoľkým podstránkam. Tieto podstránky budú obsahovať iba nadpis a paragrafy alternatívneho textu bez významu.

4.3.5 Vytvorenie kontroleru k informatívnym podstránkam

Síce sme už vytvorili vzhľad k informatívnym stránkam stále k nim nevieme pristupovať. Preto je potrebné pre nich vytvoriť kontroler. V zložke controllers teda vytvoríme triedu Pages.php, ktorého úlohou bude už spomenutý prístup k informatívnym stránkam. Táto trieda, ako máme možnosť vidieť, dedí z triedy frameworku, ktorý je nazvaná CI_Controller.

```
class Pages extends CI_Controller
{
    public function home()
    {
        $data['title'] = 'Domov';
        $this->load->view('header');
        $this->load->view('home', $data);
        $this->load->view('footer');
    }

    public function about()
    {
        $data['title'] = 'O nás';
        $this->load->view('header');
        $this->load->view('about', $data);
        $this->load->view('footer');
    }
}
```

Trieda obsahuje dve funkcie ‚home‘ a ‚about‘. V ich implementácii iba nastavíme v PHP premennú \$data do ktorej ukladáme nadpis danej podstránky a takisto načítavame jednotlivé vzhľady podstránok (home.php, about.php). Túto premennú neskôr sprístupňujeme danému vzhľadu pri jeho načítaní. V implementácii vzhľadu podstránky Domov (home.php), ktorý sme už vytvorili vložený nadpis používame ako samostatnú premennú \$title.

Po ich implementácii sme už schopní informatívne stránky, ktorých vzhľady načítavajú, otvoriť aj v prehliadači. To sme schopní spraviť prostredníctvom načítania URL odkazu `http://localhost/ci/pages/home` resp. `http://localhost/ci/pages/about`. Môžeme si všimnúť, že prvý segment obidvoch URL adries, ktorý nasleduje za základnou URL adresou (`http://localhost/ci`), ktorou pristupujeme k našej aplikácii, označuje názov nášho kontroléra. Takisto si môžeme všimnúť, že nasledujúci segment označuje už názov funkcie daného kontroléra ku ktorej pristupujeme. Pointou ako funguje prístup k stránkam vo frameworku teda je, že načítaním URL adresy pristupujeme k funkciám kontrolérov a po tomto prístupe sa vykoná úloha týchto funkcií. Tieto funkcie môžu samozrejme obsahovať aj nejaké parametre a v tomto prípade by tieto argumenty vystupovali ako ďalšie segmenty URL adresy. Tento prípad si neskôr taktiež ukážeme.

Domov Diela O nás

Domov

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas ipsum velit, consectetur eu lobortis ut, dictum at dui. Mauris dolor felis, sagittis at, luctus sed, aliquam non, tellus. Maecenas fermentum, sem in pharetra pellentesque, velit turpis volutpat ante, in pharetra metus odio a lectus. Donec vitae arcu. Curabitur bibendum justo non orci. Fusce dui leo, imperdiet in, aliquam sit amet, feugiat eu, orci. Neque porro quisquam est, qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit, sed quia non numquam eius modi tempora incidunt ut labore et dolore magnam aliquam quaerat voluptatem. Pellentesque arcu. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Curabitur bibendum justo non orci. Nulla non arcu lacinia neque faucibus fringilla. Phasellus rhoncus. Proin mattis lacinia justo. Sed convallis magna eu sem.

Následne môžeme ešte v konfiguračnom súbore `routes.php` nastaviť konfiguračný prvok `'default_controller'`, ktorého hodnotu nastavíme na `'pages/home'`. Docielime tým to, že v prípade načítania iba zakladenej URL našej aplikácie (`http://localhost/ci`), bude automaticky povolaná funkcia `'home'` v kontroléri `'Pages'`, a teda budeme pristupovať k podstránke `'Domov'`. Podstránka `'Domov'` teda v podstate bude mať funkciu domovskej stránky našej aplikácie.

```
$route['default_controller'] = 'pages/home';
```

4.3.6 Dynamická časť aplikácie

V tejto časti vývoju sa konečne dostaneme k hlavnej podstate resp. účelu frameworku CodeIgniter, a tým je vytváranie predovšetkým dynamických webových aplikácií, kedy už budeme vyvíjať aj činnosti, ktoré prebiehajú na pozadí, a ktoré užívateľ nie úplne vidí. Ide o tzv. backend. Vytvoríme teda podstránku aplikácie, v ktorej bude možné zobraziť jednotlivé kultúrne diela uložené v databáze. K tomu ich bude možné mazať a rovnako aj pridávať.

4.3.7 Zobrazenie kultúrnych udalostí

Možno ste si už v predchádzajúcej implementácii všimli, že v hlavnej navigačnej lište aplikácie sme si pripravili odkaz na podstránku `'Diela'`. Rovnako ste si možno všimli, že ako odkaz pre túto podstránku sme použili URL `http://localhost/ci/shows/index`. Touto URL sme určili, akou cestou budeme k podstránke pre zobrazovanie kultúrnych udalostí pristupovať. Z predošlých implementačných postupov už vieme, že prvým segmentom nasledujúcim za základnou časťou URL celej aplikácie je názov kontroléra, a potom ďalším segmentom je názov jeho funkcie, ku ktorej chceme pristupovať. Preto si teraz vytvoríme v zložke

```
class Shows extends CI_Controller
{
    public function index()
    {
        $data['title'] = 'Diela';
        $data['shows'] = $this->shows_model->get_shows();

        $this->load->view('header');
        $this->load->view('index', $data);
        $this->load->view('footer');
    }
}
```


controllers nový kontroler súbor Shows.php a v ňom prejdeme k implementácii funkcie index, ktorá sa bude starať o zobrazenie našich údajov z databázy o kultúrnych dielach na našej novej podstránke.

Aj v tejto funkcii nastavujeme prvky premennej \$data, ktoré neskôr načítame spoločne so vzhľadom (view) danej podstránky a tým ich budeme môcť v tomto vzhľade používať. Nastavenie nadpisu pre nás nie je nič nové, ale to nemôžeme povedať prvku shows. Do neho si uložíme pole kultúrnych diel ktoré dostaneme z našej už vytvorenej databázy. Aby sme však mohli toto urobiť, potrebujeme sa k databáze pripojiť a pomocou SQL Select dotazu dané informácie vybrať. Ako sme si povedali na začiatku tejto kapitoly, k databáze pristupujeme pomocou modelov a v tejto ukážke kódu môžete vidieť, že voláme funkciu get_shows, ktorá by nám dané pole kultúrnych diel mala vrátiť a ktorá patrí nejakej inštancii shows_model. Táto inštancia je inštanciou modelu, ktorý sme ešte neimplementovali a teda k danej implementácii rovnou prejdeme.

```
class Shows_model extends CI_Model
{
    public function __construct()
    {
        $this->load->database();
    }

    public function get_shows()
    {
        $query = $this->db->get('KulturneDielo');
        $result= $query->result_array();
        return $result;
    }
}
```

V CI je v každej triede modelu dôležité vytvoriť konštruktor, ktorý vystupuje v kóde ako funkcia a jeho úlohou je pripojenie k databáze, ktorý sa uskutoční na základe dát konfiguračného súboru database.php, ktorý sme už nastavili. To samozrejme platí v prípade ak v modeli pracujeme s databázou. Následne si môžeme všimnúť aj implementáciu funkcie get_shows, ktorú v našom kontroléri voláme. Do premennej \$query uložíme výsledok Select dotazu, ktorý sme vo frameworku vykonať pomocou funkcie get. Takáto forma tejto funkcie nám vráti všetky záznamy databázovej tabuľky KulturneDielo s ich všetkými atribútmi (stĺpcami). Pomocou funkcie result_array následne výsledok dotazu transformujeme na asociatívne pole jazyka PHP. To neskôr prostredníctvom premennej \$result vraciame ako návratový argument funkcie.

Na to aby nám komunikácia s modelom správne fungovala, je nutné tento model pridať do konfiguračného prvku model v konfiguračnom súbore autoload.php.

```
$autoload['model'] = array('shows_model');
```

Kontroler a model podstránky máme hotový, čo nám ešte chýba je ale vzhľad (view). Ten v vo funkcii index načítame ako súbor index.php a preto vytvárame takto pomenovaný súbor v zložke views. Prvou časťou tejto podstránky bude tlačidlo, po ktorého kliknutí sa dostaneme k vytváraniu nového kultúrneho diela. Pre štýlovanie opäť použijeme triedy importovaného frameworku bootstrap. Najdôležitejšou časťou je ale odkaz v ktorom je dané tlačidlo zabalené. V jeho atribúte href zadáme URL na ktorú po kliknutí nás aplikácia prepojí. Zadame iba názov segmentu ,create' ktorý sa pripojí k aktuálnemu kontroleru, a teda užívateľ bude prepojený na URL http://localhost/ci/shows/create. Funkciu k tomuto odkazu si implementujeme neskôr.

```
<a style="text-decoration: none" href="create"><button style="height: 5rem;" type="submit"
class="btn btn-success btn-lg btn-block">Vytvoriť dielo</button></a>
```

Ďalšou časťou vzhľadu podstránky je tabuľka pre zobrazenie jednotlivých kultúrnych udalostí. V ukážke kódu si ukážeme len jej najpodstatnejšiu časť a tou je telo tabuľky.


```

<tbody>
<?php foreach ($shows as $show): ?>
<tr class="table-default">
    <td style="vertical-align: middle"><?php echo $show['Nazov']; ?></td>
    <td style="vertical-align: middle"><?php echo $show['Typ']; ?></td>
    <td style="vertical-align: middle"><?php echo $show['Zaner']; ?></td>
    <td style="vertical-align: middle"><?php echo $show['Ucinkujuci']; ?></td>
    <td style="vertical-align: middle">
        <a href="<?php echo site_url('/shows/delete/' . $show['ID_KulturneDielo']); ?>">
            <button type="button" class="btn btn-outline-danger">Odstrániť</button>
        </a>
    </td>
</tr>
<?php endforeach; ?>
</tbody>

```

Riadky tabuľky, ktorá bude zobrazovať jednotlivé kultúrne diela vytvárame v cykle foreach pre ktorého implementáciu využívame jazyk PHP. V podmienke tohto cyklu využívame premennú \$shows, ktorú sme inicializovali v kontroléri ako prvok premennej \$data, ktorú sme tomuto vzhľadu následne „poslali“. Táto premenná obsahuje asociatívne pole jednotlivých riadkov tabuľky z databázy. Premenná \$show je pole, ktoré predstavuje vďaka priradeniu as v podmienke cyklu jeden riadok resp. záznam tejto tabuľky, a teda pre vypisovanie jednotlivých atribútov resp. stĺpcov záznamu využívame PHP funkciu echo a prístup k prvkom poľa \$show na základe stringu, ktorý prislúcha názvom atribútov záznamov v našej MySQL databáze. Identifikačný kľúč záznamov pre potreby informatívosti nepotrebujeme vypisovať.

Môžete si ale všimnúť, že ho využívame v poslednom stĺpci každého záznamu a to pre tlačidlo „Odstrániť“, ktorého úlohou je odstránenie záznamu, ku ktorému tlačidlo v tabuľke prislúcha. Môžete vidieť že ako odkaz URL, ktorý sa načíta po stlačení konkrétneho tlačidla je URL

http://localhost/ci/shows/delete/ID_Prisluchajuceho_KulturnehoDiela. Túto URL nám zabezpečí funkcia PHP site_url, ktorá pridá k základnej URL aplikácie segmenty zadané v argumente funkcie. Ako už určite tušíte, posledný segment tejto URL bude označovať parameter funkcie delete, ktorá bude patriť kontroleru Shows. Ako parameter sme určili ID_KulturneDielo preto, lebo ním vieme jednoznačne identifikovať každé kultúrne dielo v našej databáze.

Domov Diela O nás				
Vytvoríť dielo				
Diela				
Názov diela	Typ diela	Žáner Diela	Účinkujúci	
Ako islo vajce na vadvovku	Film	Akčný	Maroš Kramar, Elizabet Watsnova	Odstrániť
Vo štvorici po opici 1	Film	Komédia	Angelin Jolie, Morgan Freeman	Odstrániť
Obchod storočia	Predstavenie	Dráma	Jano Kolenik, Jano Folenta	Odstrániť

Ako vidíme v ukážke našej podstránky, v tabuľke už sú záznamy z databázy, ktoré sme vložili pri vytváraní našej databázy.

4.3.8 Odstránenie kultúrnych udalosti

V predchádzajúcej podkapitole sme si pripravili resp. implementovali tlačidlá pre odstránenie jednotlivých kultúrnych udalosti. Rovnako sme si tam aj rozobrali URL adresu, ktorou sa po kliknutí na tieto tlačidlá

načíta. Preto teraz v kontroleri Shows implementujeme funkciu delete, ktorá bude už v tomto prípade prijímať aj jeden argument, ktorým bude identifikačné číslo kultúrneho diela v databáze.

```
public function delete($id)
{
    $this->shows_model->delete_show($id);
    redirect('shows/index');
}
```

Môžete vidieť, že najdôležitejším riadkom je povolanie funkcie delete_show, ktorá patrí modelu Shows_model. Následne po vykonaní už iba presmerujeme užívateľa na podstránku s výpisom jednotlivých kultúrnych diel a to prostredníctvom funkcie index v tom isto kontroleri. Na začiatku kapitoly sme si povedali, že zatiaľ čo kontroleri sú vo fremowrku povinné modely a vzhľady (views) sú voliteľné. Prípad kedy sme implementovali kontrolér so vzhľadmi sme už mali pri kontroleri Pages.php. Teraz sme sa dostali k prípadu kde používame kontroler (jeho funkciu delete), model ale nepoužívame žiadny view. Pri procese vymazania záznamu z databázy nie je nutné niečo zobrazovať užívateľovi, maximálne tak hlásenie o úspešnosti vykonania tohto úkonu.

Následne si ukážeme implementáciu funkcie delete_show v modeli Shows_model. Pre vymazanie záznamu používame frameworkom vstavanú funkciu delete, ktorou úlohou je práve mazanie záznamov z jednotlivých tabuliek. V prvom parametri zadávame názov tabuľky a v druhom pole, ktorým na základe názvu atribútov tabuľky identifikujeme záznamy, ktoré chceme vymazať. V našom prípade identifikujeme tento záznam podľa identifikačného čísla záznamu, ktorý sme obdržali ešte ako parameter funkcie delete v kontroleri.

```
public function delete_show($id)
{
    $this->db->delete('KulturneDielo', array('ID_KulturneDielo' => $id));
    return true;
}
```

4.3.9 Pridanie kultúrnych udalostí

Na vytváraní podstránky pre zobrazovanie udalostí sme si hovorili, že k implementácii pridávania nových kultúrnych udalostí, pre ktoré sme si pripravili tlačidlo s prepojením na URL <http://localhost/ci/shows/create>, sa vrátíme neskôr. Tejto implementácii sa budeme venovať v tejto časti. Ako prvé si vytvoríme nový vzhľad (view), ktorý bude obsahovať formulár pre vytvorenie nového diela. Tento súbor teda pomenujeme create.php a umiestnime ho do zložky views.

Dôležitou súčasťou implementácie je po prvé vypísanie výsledku funkcie validation_errors, ktorý v prípade zlom vyplnení formulára od užívateľa vypíše jednotlivé chyby pred formulárom. Po druhé je dôležité zavolať funkciu pre otvorenie (form_open) a uzavretie (form_close) formulára, ktoré obaľujú implementáciu formulára. V prípade funkcie form_open je dôležité ako parameter funkcie uviesť cestu k funkcii kontroleru, ktorá vstupy z formulára bude spracovávať, a teda ktorej sa tieto vstupy majú poslať. V našom prípade ide o cestu ,shows/create'.

Dôležitou je aj samotná implementácia prvkov formulára. Ten by mal v prvom rade obsahovať jednotlivé vstupy pre užívateľa. Ich dôležitou súčasťou je atribút ,name', prostredníctvom ktorého hodnotou budeme v kontroleri pristupovať k jednotlivým vstupom od užívateľa. Nemôžeme zabudnúť ani na tlačidlo, ktoré musí byť typu submit, čím sa zabezpečí že po jeho kliknutí budú spracované vstupy od užívateľa.

```

<label for="name">Názov</label>
<input type="text" class="form-control" name="name" placeholder="Vložte názov diela">
<small id="firstnameHelp" class="form-text text-muted">Maximálne 30 znakov</small>

<div class="row">
    <div class="col-sm-6 col-form-label form-group">
        <label for="type">Typ</label>
        <input type="text" class="form-control" name="type" placeholder="Vložte typ diela">
        <small id="firstnameHelp" class="form-text text-muted">Maximálne 30 znakov</small>
    </div>
    <div class="col-sm-6 col-form-label form-group">
        <label for="genre">Žáner</label>
        <input type="text" class="form-control" name="genre" placeholder="Vložte žaner diela">
        <small id="firstnameHelp" class="form-text text-muted">Maximálne 30 znakov</small>
    </div>
</div>

<label for="text">Učinkujúci</label>
<textarea type="text" class="form-control" name="text" placeholder="Vložte účinkujúcich"></textarea>
<small id="firstnameHelp" class="form-text text-muted">Maximálne 250 znakov</small>

<button type="submit" style="margin-top: 20px" class="btn btn-primary btn-lg btn-block">Potvrdiť</button>

```

Môžeme teda prejsť k implementácii funkcie create v kontroléri Show.php. V prvej časti tejto funkcie sú pomocou vstavanej funkcie set_rules nastavené pravidlá pre jednotlivé elementy formulára ktorý sme už implementovali. Táto funkcia vyžaduje ako prvý parameter identifikačný názov elementu, ktorý sme vkladali ako hodnotu atribútu name pri vytváraní formulára. Ďalším parametrom je ľubovoľné pomenovanie elementu, ktoré sa použije pri vypísaní chyby vyvolanej zlým vstupom užívateľa, a posledným parametrom je už samotné pravidlo. V našom prípade sme použili pravidla ,required', čo znamená že vstup je v danom elemente formulára vyžadovaný a teda nemôže ostať nevyplnený.

```

public function create()
{
    $this->form_validation->set_rules('name', 'Nazov', 'required');
    $this->form_validation->set_rules('type', 'Typ', 'required');
    $this->form_validation->set_rules('genre', 'Žáner', 'required');
    $this->form_validation->set_rules('text', 'Učinkujúci', 'required');
}

```

Ďalšou časťou funkcie je podmienka ,if'. Tá skontroluje pomocou funkcie ,run', či sú vstupy od užívateľa zadane podľa jednotlivých pravidiel ktoré sme implementovali vyššie. V prípade že nie, funkcia vráti hodnotu ,false' a prejde do prvej vetvy podmienky. Podmienka bude takto vyhodnotená aj v prípade, že sa do našej funkcie ,create' pristúpilo z podstránky pre vypísanie kultúrnych diel (funkcia index) a to kliknutím na tlačidlo ,Pridať dielo'. V tejto vetve sa načíta vzhľad (view) create.php, ktorý obsahuje formulár, a ktorý sme pred chvíľou implementovali. V prípade nesplnenia pravidiel formulára sa vypíšu chyby.

```

if($this->form_validation->run() == False)
{
    $data['title'] = 'Vytvorenie nového diela';

    $this->load->view('header');
    $this->load->view('create', $data);
    $this->load->view('footer');
}

```

V prípade, že vstupy od užívateľa budú v poriadku pristúpi sa k vetve podmienky ,else'. V tej najprv uložíme vstupy od užívateľa do rôznych premenných a to prostredníctvom vstavanej funkcie frameworku ,input->post'. Jej parametrom sú opäť hodnoty atribútu ,name' jednotlivých vstupných prvkov formuláru. Následne premenné s uloženými vstupmi použijeme ako parametre našej funkcie create_show, ktorú implementujeme našom modeli Shows_model. Po jej vykonaní už povoláme len funkciu ,redirect', ktorou bude užívateľa prepojený opäť na podstránku pre vypísanie našich kultúrnych diel.

V modeli teda spomenutú funkciu s názvom ,create_show' implementujeme. Jej obsahom je najprv vytvorenie asociatívneho poľa s názvom \$data, v ktorom priradíme jednotlivé vstupy od užívateľa k reťazcom, ktoré sa zhodujú s názvami atribútov tabuľky v našej databáze. Následne pomocou funkcie ,db->insert' vožime záznam do databázy. Parametrami funkcie sú názov tabuľky v databáze a takisto pole s hodnotami atribútov záznamu, ktorý chceme vytvoriť.

```
public function create_show($name, $type, $genre, $text)
{
    $data = array(
        'Nazov' => $name,
        'Typ' => $type,
        'Zaner' => $genre,
        'Ucinkujuci' => $text
    );

    $this->db->insert('KulturneDielo', $data);
    return true;
}
```

Týmto sme pridávanie nových kultúrnych diel dokončili. Pridanie kulturného diela môžeme vidieť v ďalšej ukážke.

[Domov](#) [Diela](#) [O nás](#)

Vytvorenie nového diela

Názov

John Wich 2

Maximálne 30 znakov

Typ

Film

Maximálne 30 znakov

Žáner

Akčný

Maximálne 30 znakov

Učinkujúci

Tomy Harry, Angelina Jolie

Maximálne 250 znakov

Potvrdiť

Dielo

Názov diela	Typ diela	Žáner Diela	Účinkujúci	
Ako islo vajce na vandrovkú	Film	Akčný	Maroš Kramár, Elizabet Watsnova	Odstrániť
Vo štvorici po opici 1	Film	Komédia	Angelín Jolie, Morgan Freeman	Odstrániť
Obchod storočia	Predstavenie	Dráma	Jano Koleník, Jano Folenta	Odstrániť
John Wick 2	Film	Akčný	Tomy Harry, Angelína Jolie	Odstrániť

5 Záver

V tomto článku sme si ukázali, ako jednoduché je vytvoriť dynamickú webovú aplikáciu vo frameworku CodeIgniter. V prípade, že by ste chceli sa základy tohto frameworku naučiť lepšie odporúčame tutorial, v ktorom autor vytvára komplexnejšiu aplikáciu a z ktorého sme niekoľko informácií čerpali aj my. Tento tutorial je dostupný na tomto odkaze:

https://www.youtube.com/watch?v=I752ofYu7ag&list=PLlilGF-RfqbaP_71rOyChhjeK1swokUIS.

6 Bibliografia

1. [Online] <https://cs.wikipedia.org/wiki/CodeIgniter>.