FAKULTA
APLIKOVANÝCH VĚD
ZÁPADOČESKÉ
UNIVERZITY
V PLZNI

students.kiv.zcu.cz

KIV/WEB - Semestrální práce

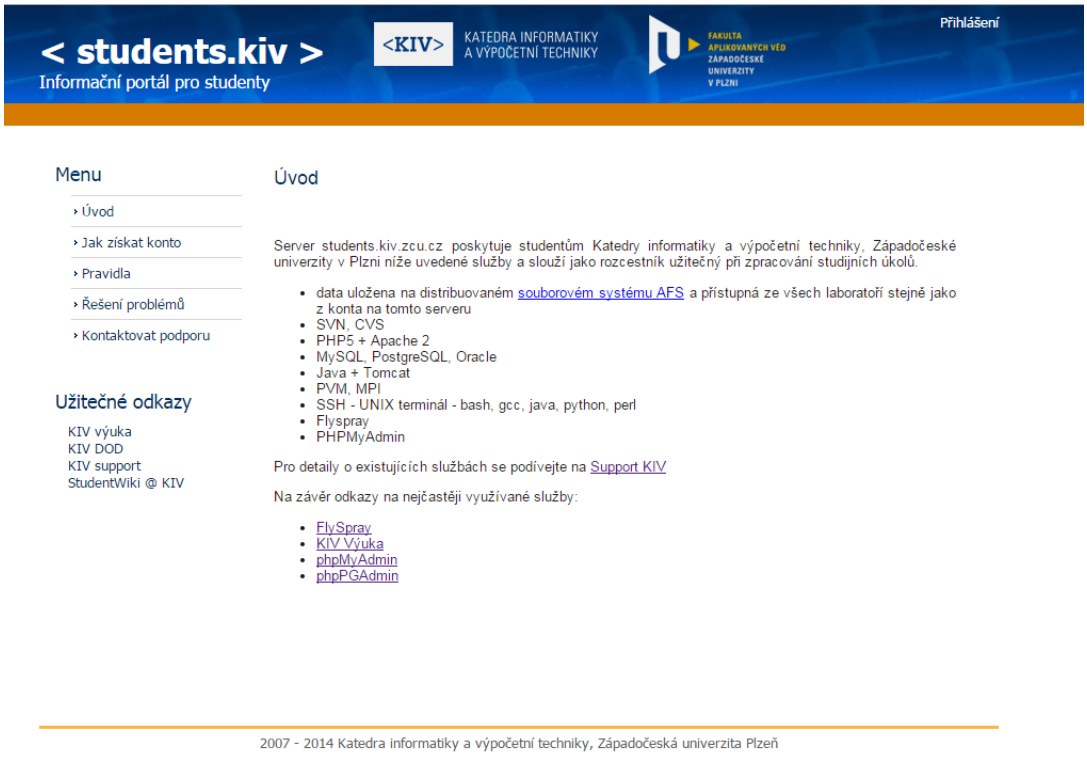
student:	*Petr Kukrál*
studijní číslo:	*A13N0112P*
email:	kukral@students.zcu.cz
datum:	1. 12.2014

Zadání

Zadáním semestrální práce je vytvoření nových webových stránek students.kiv.zcu.cz. Stránky budou mít jedno levé menu, které nepůjde zanořovat. Budou editovatelné a ke každé stránce bude možné připojit soubory. V layoutu stránky bude seznam odkazů, který také bude možné editovat. Veškeré editace bude provádět administrátor, který musí být zaregistrován v systému. Stránky budou obsahovat kontaktní formulář, který odešle obsah formuláře na danou e-mailovou adresu. Přihlašování na stránky bude probíhat přes kerberos zču loginem. Stránky musí využívat pretty url. Stránky by měly být především funkční a obsahovat co nejméně CSS3 a javascriptů, které by zajišťovaly hezčí vzhled.

Vzhled

Vzhled těchto stránek vychází z webu kiv.zcu.cz. Vzhledem k tomu, že je web kiv.zcu.cz rozsáhlejší, než web students.kiv.zcu.cz, vzhled bylo nutné zjednodušit jak je vidět na obr. 1. Na administraci byl použit framework bootstrap z důvodu jednoduchosti a přehlednosti.



obr.1 - vzhled stránek

Systém

V práci nebylo povoleno použití žádného PHP frameworku. Proto bylo potřeba vytvořit vlastní miniaplikaci, která by stránky obsluhovala. Celá aplikace je napsána v MVC architektuře. Obsahuje tři následující části:

- Modely, které představují jednotlivá DAO.
- View, který představuje jednotlivé náhledy stránek a layout.
- Controlery.

Životní cyklus aplikace

Webová aplikace volá soubory v určitém pořadí. Prvním klíčovým souborem je bootstrap.php, který vytváří routy, spouští aplikaci a zajišťuje autoloading tříd. Druhým klíčovým souborem je application.class.php. Ten inicializuje všechny pomocné třídy (jako je např. třída na práci se session). Dále vytvoří správný controler podle třídy Router a postupně spouští jeho metody. Životní cyklus tedy probíhá takto:

- index.php
- bootstrap.php
- aplication.class.php
- controler - vybere se controler podle routy
- view - vybere se view podle routy

Cyklus controleru

U controleru se spouští několik metod, ve kterých můžeme ve správný čas ovlivnit běh aplikace. Jsou to tyto metody (spouštěné v tomto pořadí):

- startUp()
- action($pageName) nebo actionPageName
- checkDoParam()
- render($pageName) nebo renderPageName

Metoda startUp je jediná pro celý controler. Vzhledem k tomu, že se spouští první, je vhodné v ní kontrolovat například přístupová práva.

Aplikace s metodami action a render pracuje následovně. Nejdříve se aplikace snaží najít metody actionPageName a renderPageName, kde PageName představuje řetězec za posledním lomítkem v adrese. Pokud je nalezne, vykoná obslužný kód v těchto metodách. Tyto metody jsou užitečné v případě, že chceme reagovat na každou url zcela odlišně, jako je to například v administraci. Pokud je nenalezne, spustí metody action a render a předá jim

název stránky. Tento způsob použijeme v případě, že chceme na všechny stránky reagovat podobně a chceme např. jen načítat jiná data z databáze, jako je to nyní na frontendu.

Metoda checkDoParam je jedna pro celý controler. Tato metoda kontroluje, zda má aplikace vykonávat nějakou specifickou funkci, například zpracovat odeslaný formulář, nebo odhlásit uživatele. V případě, že má aplikace vykonávat tuto specifickou funkci, je nastaven v adrese atribut “do”, ve kterém je název označující danou činnost. V základu všechny controlery umí reagovat na oba dva stavy. Při odhlášení uživatele se předá v parametru “do” řetězec “sign-out” a při odeslání formuláře řetězec “submit-form”. Více informací o zpracování formulářů je v kapitole Formuláře. Pokud by bylo nutné dopsat si vlastní reakci na parametr “do”, stačí překrýt metodu checkDoParam, jako je to například v AdminControleru. Vždy je ale nutné spouštět i jeho rodiče, jinak by aplikace nefungovala správně.

Formuláře

Ve view se formulář napíše klasicky v html. Action pak nasměrujeme na controler, který má formulář zpracovat a přidají se parametry ?do=submit-form&form=[NazevFormulare]. Parametr “do” s nastavením “submit-form” controleru oznámí, že se má obsloužit odeslaný formulář. Parametr “form” nám pak říká, který formulář se má obsloužit.

Na straně controleru se pak zavolá metoda s názvem submit[NazevFormulare], která obslouží samotné odeslání formuláře. Nemusíme se tak nijak starat o to, aby byla spuštěna správná metoda čtením parametrů z url. To už aplikace udělá za nás.

Přesměrování

Ve view se do odkazu píše adresa tak, jak ji vidí uživatel v prohlížeči. Pokud ale chceme přesměrovat stránku uvnitř controleru (například když chce vstoupit do administrace nepřihlášený uživatel), použijeme metodu redirect(). Ta nás správně přesměruje a okamžitě ukončí další vykonávání aplikace.

Metoda redirect má tři parametry, ale pouze první je povinný:

- název controleru
- název view
- GET parametry, co se mají poslat

Název controleru je zde název třídy. Pro front-end to nyní bude “FrontControler”. Metoda redirect pak využije zaregistrovaných rout, aby vytvořila správný link. V případě Front Controleru to bude link “/názevStránky” a v případě Admin Controleru to bude link “/admin/názevStránky”. Pokud by jsme tedy chtěli časem zpětně přejmenovat, na jaké stránce se nám zobrazuje Front Controler z “/” na “/front”, stačí změnit tuto informaci u registrace routy a všechny odkazy přes metodu redirect začnou okamžitě přesměrovávat na novou url.

Pokud chceme pomocí metody redirect jen obnovit stránku, na které se nacházíme, stačí ji do prvního parametru předat slovo “this”. Metoda si pak sama zjistí, v jakém Controleru a na jaké view se nachází a druhý parametr s názvem view ignoruje. Třetí parametr metody funguje správně, tedy si touto cestou lze obnovit stránku a přidat, nebo ubrat nějaké parametry GET. Standardní použití je například po zpracování formuláře, kdy chceme stránku obnovit, ale smazat parametr “do”, který nám vyvolal obsluhu odeslání formuláře.

Pretty url

V celé aplikaci fungují Pretty url. Pomocí rout můžeme přidat neomezeně controlerů. Název view si už řeší každý controler sám. Routy se registrují v bootstrapu a zpracovává je třída Router. Pokud by jsme chtěli přidat další routu např. na TestControler s adresou “/test/názevStránky”, uděláme to takto:

Do bootstrapu u registrace rout přidáme další řádek

$container->addRoute(new Route("/test/", "TestControler "));

Od této chvíle již budou adresy fungovat správně.

Třída User

Tato třída umožňuje přihlásit a odhlásit uživatele. Udržuje také informaci o tom, zda je uživatel přihlášen a jestli je administrátor. Ke třídě se dostaneme z každého controleru pomocí $this->user.

Třída Messages

Aby jsme nemuseli psát vlastní obsluhu na vypsání každé zprávy uživateli, stačí použít třídu Messages. Vrátíme si ji v controleru pomocí $this->messages a v layoutu je uložená v proměnné $messages. Této třídě pomocí metody addMessage() předáme zprávy, které se mají vypsat. Tyto zprávy se pak v layoutu vypíší. Pokud bychom si chtěli napsat vlastní layout, bylo by nutné napsat obsluhu, která zprávy vypíše. Zprávy se smažou až v momentě, kdy se uživateli opravdu zobrazí. Tedy až v momentě dokončení metody render. Kdyby jsme například přesměrovali stránku z metody action, zprávy se v aplikaci podrží a vypíší se až po přesměrování.

Závěr

Aplikace je záměrně napsaná tak, aby bylo možné rychle přidávat další stránky s co možná nejmenším úsilím, s již funkční pretty url, posíláním zpráv a pod. Aplikace sama řeší mnoho věcí vnitřně, jako je například zavolání správné metody na obsluhu odeslaného formuláře, či práci se session pomocí třídy Session. Nijak ale neovlivňuje původní globální proměnné. Tedy pokud by si chtěl programátor napsat obsluhu formuláře sám, aplikace mu v tom nijak nebrání.