Portál pre odborné publikovanie ISSN 1338-0087

PRADO framework

Liner Lukáš · Informačné technológie, Študentské práce

08.02.2013

PRADO framework je objektovo orientovaný framework, určený na rýchly vývoj webových aplikácií v programovacom jazyku PHP 5. PRADO framework je určený pre stredne pokročilých a pokročilých vývojárov v jazyku PHP. Pre zvládnutie tohto frameworku je nutné ovládať aspoň základné princípy jazykov HTML, XHTML, CSS, JavaScript, AJAX.

Framework oddeľuje prezentačnú vrstvu od aplikačnej logiky. Taktiež poskytuje široké spektrum vlastných webových komponentov, akými sú napríklad autorizačné a autentifikačné moduly, cache moduly, validátori, šablóny, komunikáciu s databázou a taktiež podporuje možnosť tvorby vlastných modulov a algoritmov. Samozrejme je možné upravovať podľa vlastných potrieb už existujúce moduly. Tento framework vám môže urýchliť a zjednodušiť vývoj aplikácií, pretože nie je nutné do detailov ovládať princípy programovania v hore uvedených jazykoch. Tvorcovia frameworku sa zamerali hlavne na bezpečnosť a dodržiavanie princípov (aktuálnych trendov) pri tvorbe frameworku. Na internete je dostupná detailná dokumentácia a taktiež existuje fórum na ktorom sa môžete pýtať na svoje otázky. Framework je možné stiahnuť z adresy http://www.pradosoft.com, kde sa dozviete aj bližšie informácie ktoré vás zaujímajú.

PRADO framework (PHP Rapid Application Development Object-oriented)

PRADO framework je objektovo orientovaný framework, ktorý je určený pre rýchly vývoj webových aplikácií v programovacom jazyku PHP verzia 5. PRADO poskytuje širokú škálu funkcii ako sú:

- Možnosť objektovo orientovaného programovania
- Opakovateľne využiteľný kód
- Event-driven (udalosťami riadené) programovanie
- Oddelenie prezentačnej vrstvy od aplikačnej logiky
- Konfigurovateľná a zásuvne modulárna architektúra
- Kompletné spektrum databázovej podpory
- Funkcie určené pre webové komponenty HTML ako vstupné kontroly, validátory, DataGrid, šablóny, …
- Využitie AJAXU vo webových komponentoch
- Lokalizáciu a zachytenie chýb/výnimiek
- Generické cache moduly a selektívne výstupy do medzi pamäte

- Rozšíriteľné autentizačné a autorizačné rámce
- Bezpečnostné opatrenia, prevencia (XSS)
- Dodržanie validného XHTML formátu
- Kvalitná dokumentácia

Tento framework je voľne šíriteľný. Dokumentáciu, informácie a podrobnosti môžete nájsť na internetovej stránke frameworku www.pradosoft.com.

Inštalácia frameworku

Inštalácia je veľmi jednoduchá. A skladá sa z 4 jednoduchých krokov. Nevyhnutná je inštalácia apache (xamp, lamp, wamp), pred začatím inštalácie frameworku. Postup inštalácie:

1. Je potrebné stiahnúť framework PRADO z adresy: http://www.pradosoft.com/download/
2. Rozbalíme framework do priečinku (v našom prípade ../htdocs)
3. Tým sme framework nainštalovali
4. Ďalšími možnosťami je jeho konfigurácia, nastavenia si môžeme pozrieť localhost/prado/requirements, kde dostanete nasledovnú tabuľku

Tabuľka frameworku PRADO requierments s nastaveniami, rozšíreniami a povoleniami

Vytvorenie novej aplikácie

Vytvoriť novú aplikáciu je možné dvomi spôsobmi. Prvým spôsobom je vytvorenie celej štruktúry manuálne. Tento spôsob nie je najvhodnejší. Lepším spôsobom je použiť prado-cli.bat ktorý je určený na operácie tohto typu. Len pripomenieme, že pred vytvorením novej aplikácie je potrebné mať framework funkčný na našom PC. Postup inštalácie novej aplikácie je zachytený v nasledovných krokoch:

1. Spustíme Apache
2. Otvoríme si comand line “cmd” príkazový riadok
3. Do príkazového riadku napíšeme `path/to/framework/prado-cli.php –c MojWeb`. Ak chceme informácie o možnostiach použijeme namiesto `“-c MojWeb“ “-h“`. Pričom –c predstavuje vytvorenie nového projektu a MojWeb predstavuje názov projektu
4. Tým sa nám v priečinku htdocs vytvoril nový adresár MojWeb v ktorom je kostra nášho nového projektu

Mapovanie databázy

Mapovanie realizujeme tiež najlepšie pomocou prado-cli.php, ale pred tým ako môžeme začať mapovať databázu musíme mať vytvorené funkčné pripojenie na databázu ktoré sa vytvára v súbore application.xml a vyzera nasledovne:

```
<module id="db" class="System.Data.TDataSourceConfig">
<database Charset="utf8" password="heslo" username="root"
ConnectionString=" mysql:host=localhost;dbname=test"/>
</module>
```

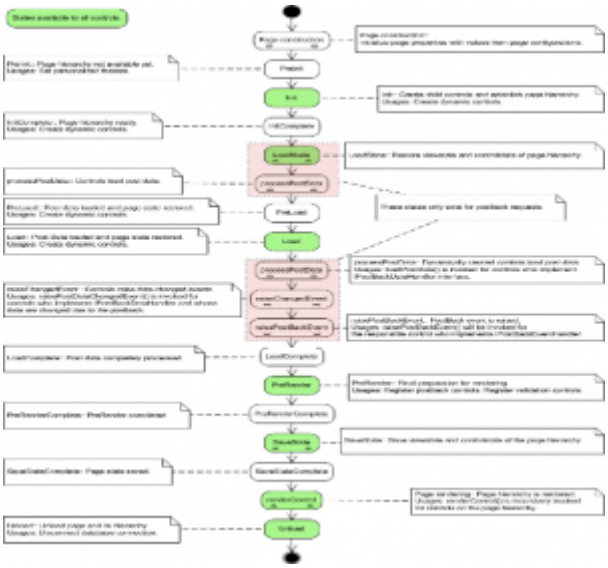
Keď máme vytvorené spojenie s databázou môžeme vytvoriť jej mapovanie a to nasledovne:

1. Nastavíme sa do koreňového adresára nášho projektu pomocou príkazu `cd /pathToRoot/MojWeb`
2. Spustíme `path/to/framework/prado-cli.php shell`.
3. Následne môžeme generovať mapovania na jednotlivé tabuľky pomocou príkazov `generate table Application.database.TableRecord` pričom table predstavuje názov tabuľky v databáze, database predstavuje priečinok obsahujúci triedy maovanej databázy a TableRecord predstavuje názov triedy obsahujúcej mapovanie tabuľky table.

Prezentačná vrstva a aplikačná logika v PRADE

Ideou vývojárov frameworku bolo vytvoriť framework, ktorý by mal oddelenú aplikačnú logiku od prezentačnej vrstvy. Aplikačnou logikou môžeme nazvať aj biznis vrstvu. Má za úlohu spracovávať dáta a komunikovať s prezentačnou a dátovou vrstvou. V PRADE sú najvýznamnejšie tieto štruktúry:

Dokumenty typu page – Pomocou nich tvoríme prezentačnú vrstvu (jednotlivé stránky) a to konkrétne nahrádzajú dokumenty (html, xhtml), ktoré sú bežne používané pri programovaní v php-čku bez použitia frameworku. Dokumenty typu page obsahujú všetky základné značky, ktoré obsahujú aj html a xhtml a sú rozšírené o vlastné značky, ktoré majú špecifický význam. Takáto štruktúra tvorí základ celého frameworku. Pomocou nich je schopný koncový užívateľ posielať požiadavky a prijímať odpovede. Na obr. č. 1 je znázornený životný cyklus stránky od jej načítania až po jej opustenie.



Obr.1: životný cyklus stránky

Dokumenty typu php – Zabezpečujú aplikačnú logiku zobrazujúcu na stránkach.

Dokument typu php musí mať rovnaké meno ako jeho prislúchajúci dokument typu page. Napríklad: home.php – home.page. Tieto dokumenty nie sú schopné existencie jeden bez druhého.

Dokumenty typu tpl – V podstate majú rovnaký význam ako page dokumenty s tým rozdielom, že ich štruktúru sme schopný použiť neobmedzený počet krát bez nutnosti prepisovania kódu. Používajú sa hlavne na layout stránok, alebo na vytváranie zásuvných štruktúr portlets.

Zásuvné štruktúry portlets – Každá takáto štruktúra musí obsahovať aplikačnú časť a prezentačnú časť(dokumenty tpl a php). S ich pomocou sme schopný niektoré štruktúry naprogramovať raz a využívať neobmedzený počet krát. Takéto štruktúry nie sú schopné existovať samostatne, môžeme ich len zasúvať do iných dokumentov. Napríklad pomocou príkazu <com:Application.CestaKStrukture.NazovStruktury />

Špeciálne dokumenty:

- application.xml – slúži na konfiguráciu celej aplikácie. Obsahuje špecifikáciu ciest (path) iných objektov, menný priestor (namespace), konfiguráciu modulov služieb (service) a parametre.
- config.xml – slúži na vytvorenie prístupových pravidiel. Špecifikuje kto a kde má povolený alebo zakázaný prístup. Napríklad: Prihlásený užívateľ má iné prístupové práva ako neidentifikovaný používateľ. config.xml nevytvára roly len udáva, aké sú prístupové práva jednotlivých rol v systéme.

Autentizácia a autorizácia implementovaná frameworkom

Autentizácia – jedná sa o proces overovania pri ktorom sa zisťuje, či je niekto tým za koho sa vydáva. Zvyčajne sa tu overuje správnosť údajov používateľské meno a heslo, ale môžu to byť aj iné spôsoby preukázania totožnosti. Napríklad čipové karty…

Autorizácia – v tomto prípade sa jedná o dynamické zisťovanie, či osoba ktorá je autentizovaná má prístupové práva pre prácu s určitými zdrojmi. Pravidlá sú zadefinované vo vyššie uvedenom dokumente config.xml.

PRADO plne podporuje prácu a podporu pri použití autentizačných a autorizačných modulov. Túto funkcionalitu zaisťujú dva základné moduly, TAuthManager používaný v kombinácii s TUserManager, ktorý implementuje možnosť čítania z lokálnej databázy (TDbUserManager). V prípade, že by funkcie, ktoré tieto moduly poskytujú neboli dostatočné je možné si ich rozšíriť, alebo vytvoriť úplne nové.

Práca s lokálnou databázou

Práca s databázami na úrovni php je dosť zdĺhavá oproti možnostiam, ktoré poskytujú frameworky. Aj tento framework poskytuje podporu pre prácu s databázami. Sú dve možnosti pre ktoré sa môžeme rozhodnúť, buď použijeme Active Record alebo Data Mapper. Avšak je medzi nimi dosť veľký rozdiel.

Active Record – Je objekt určený na základné operácie s databázou. Tento objekt taktiež zahŕňa základnú logiku pre nadviazanie spojenia. Práca je veľmi rýchla a

jednoduchá ako vidíme na obr. č. 6. Active Record poskytuje funkcionalitu pre nasledovné úlohy:

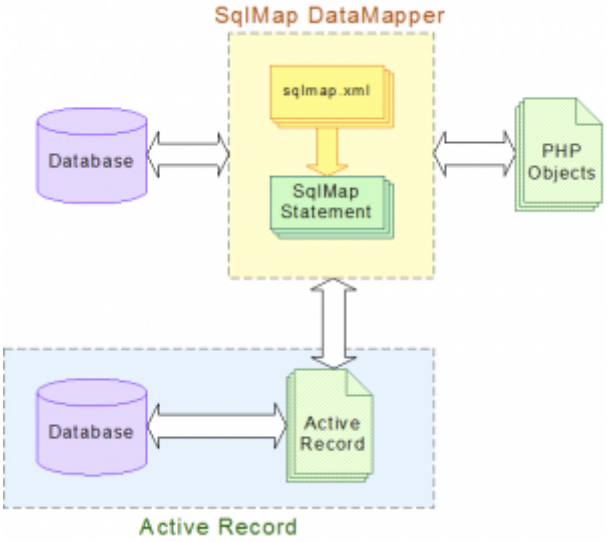
- čítanie, upravovanie, mazanie záznamov v databáze
- metódu na vyhľadávanie pomocou základných SQL dotazov do ktorej vložíme údaje, ktoré chceme vyhľadať a tá nám vráti objekt s požadovanými dátami
- kardinalita – použitie vzťahov (súvisiacich so vzťahmi) “has many” 1:N, “has one” 1:1, “belongs to” N:1 a “many to many” N:M medzi prepojenými tabuľkami
- takzvané lazy loading vzťahov, jedná sa o spôsob pri ktorom sa nevracajú dáta o ktoré sme nepožiadali, odpoveďou sú len dopytované dáta

```
class OblastRecord extends TActiveRecord
{
 const TABLE='table_name'; //názov tabuľky
 public $id; // názov riadku tabuľky
 public $value; // názov riadku tabuľky

 // metóda určená na vyhľadávanie
 public static function finder($className=__CLASS__) {
  return parent::finder($className);
 }

 public static $RELATIONS=array
 (
 'oblasti' => array(self::MANY_TO_MANY, 'AnotherTable'),
 // Vzťah N:M
 );
}
```

Trieda reprezentujúca tabuľku table_name definovaná pomocou Active Record.

Problém nastáva, keď nám už nestačia len základné riadkové operácie a radi by sme využili vlastné SQL príkazy pre efektívnejšie získavanie údajov, pretože náš projekt narástol do veľkých rozmerov a my potrebujeme efektívne pracovať s dátami. Pomocou Active Record by bola táto práca veľmi zložitá, časovo náročná a chaotická. V takomto prípade nastupuje Data Mapper.

Data Mapper – Používa sa pri väčších projektoch. Oddeľuje štruktúru SQL dopytov od miesta a spôsobu akým a kde sú vykonané. Môžeme si vytvárať dopyty podľa vlastných požiadaviek a nie sme ničím obmedzovaný. Poskytuje plnú kontrolu nad databázou. Active Record a Data Mapper je možné medzi sebou aj kombinovať. Vzťah medzi nimi je znázornený na obr. č. 2.



Obr.2: Vzťah medzi Active Record a Data Mapper.

Webové komponenty v PRADO frameworku

PRADO obsahuje všetky značky, ktoré obsahuje html dokument, ale navyše obsahuje aj sadu vlastných značiek, ktoré majú rôznu funkcionalitu. Niektoré z nich si ukážeme.

Validátory – Slúžia na ošetrovanie vstupov do formulára. Môžu byť statické a dynamické a môžu ošetrovať vstupy rôznych typov. Princíp ich činnosti je vo všeobecnosti takýto: pri zadaní nejakej hodnoty skúsia, či je hodnota korektná, ak je tak potvrdia korektnosť údaja, ak nie tak oznámia nesprávnosť vstupu a vypíšu chybovú hlášku a zamedzia odoslaniu formulára. V určitých prípadoch sa môže ich činnosť líšiť. Medzi tieto validátory patria aj:

- TActiveCustomValidator – validácia vstupov na strane servera
- TCompareValidator – slúži na porovnávanie dvoch hodnôt
- TRequiredFieldValidator – zisťuje, či bola zadaná hodnota
- TRegularExpressionValidator – zisťuje správny formát údajov
- TEmailAddressValidator – zisťuje správnosť emailovej adresy
- TDataTypeValidator – zisťuje, či bol zadaný správny dátový typ

„Databázové značky“ – jedná sa o značky, ktoré nie sú v html definované, ale v PRADO frameworku sú zadefinované. V niektorých situáciách je to veľká výhoda. Existuje veľké množstvo takýchto značiek, uvediem len jeden príklad značky, ktorá je často využívaná. Zoberme si dáta z databázy, ktoré chceme vkladať do tabuľky. Túto tabuľku musíme naplniť pomocou nejakej funkcie, alebo ručne, čo môže trvať značnú chvíľu. Tu môžeme využiť značku <com:TDataGrid>. Použitie tejto značky vidíme v nasledovných útržkoch kódu.

```
<com:TDataGrid
 ID="members"
 <!– automatické generovanie tabuľky -->
 AutoGenerateColumns="true"
>
```

Značka reprezentujúca dáta z tabuľky members v databáze (dokument.page)

```
public function onInit($param)
{
 parent::onInit($param);
 $members = new MembersRecord();
 //vyhľadanie v databáze
 $members = MembersRecord::finder()->findAll();
 // zdroj informácií
 $this->members->DataSource = $members;
 // naplnenie tabuľky
 $this->members->dataBind();
}
```

Vloženie hodnôt z databázy do značky s ID members (dokument.php)

Vo vyššie uvedenom kóde sme vytvorili tabuľku na stránke do ktorej sme následne vložili záznamy členov z databázy, ktorých sme si vyhľadali pomocou triedy MembersRecord, ktorá má implementovanú metódu na vyhľadávanie. V databáze by mohlo byť záznamov aj milión všetko, by bolo automaticky vykonané za pomoci týchto pár riadkov.

„Html inputy“ – Tieto značky sú v html už definované v PRADO frameworku bola ich funkcionalita rozšírená o niektoré užitočné vlastnosti. Medzi tieto značky patria napríklad:

- TLiteral – reprezentuje statický text, ktorý vieme jednoducho zmeniť
- TTextBox – vstup textu, ktorý sa dá ľahko zvalidovať validátormi
- TButton – reprezentuje <input type=“button“/> veľa nových možností
- TDropDownList – vyrolovacie menu s možnosťou automatického odoslania
- THyperLink – je kombináciou značiek
- TImage – značka reprezentuje značku z html
- TFileUpload – slúži na posielanie súborov na server
- TDatePicker – kalendár, je to dobre formátovaný vstup pre zadávanie dátumu
- TRepeater – zobrazenie údajov rovnakej štruktúry

TRepeater – Jeho úlohou je zobrazovať dáta, ktorých štruktúra sa opakuje viackrát. Takéto dáta sú vložené do DataSource, ktorý tvorí zdroj informácií, z ktorých TRepeater (ďalej len repeater) čerpá. Keď je použitá metóda dataBind() repeater vytvorí toľko riadkov, koľko záznamov je vložených do DataSource. Každý z týchto riadkov je ovládateľný cez items vlastnosti. Ďalej repeater môže obsahovať hlavičku a pätu (<prop:HeaderTemplate>, <prop:FooterTemplate>) v ktorých je popis stĺpcov. Každý jeho riadok reprezentuje šablóna v ktorej je definovaný vzhľad a štruktúra.

Existujú dva druhy šablón (<prop:IItemTemplate>, <prop:AlternatingItemTemplate>), pričom prvá reprezentuje každý nepárny riadok a druhá každý párny riadok. Od verzie 3.1.0 je možné do repeater vkladať navyše renderer, ktorý predstavuje šablónu zložitých riadkov, ktoré chceme do repeatra vložiť, aby sme štruktúru, ktorá je definovaná v renderery nemuseli viackrát opätovne písať. V prípade zavolania metódy dataBind(), repeater vykoná nasledovné úlohy pre každý riadok:

Vytvorí item založený na šablóne alebo renderery

- Naplní ho dátami
- Vyvolá udalosť OnItemCreated
- Pridá item ako ovládateľnú položku
- Zavolá metódu dataBind() na jednotlivé items
- Vyvolá udalosť OnItemDataBound

K jednotlivým položkám môžeme pristupovať dvoma spôsobmi. Prvý spôsob je, že definujeme v položke metódu OnItemCommand, ktorá je jedinečná pre každý item. Alebo druhý spôsob je, že použijeme v definícii repeatra parameter DataKeys, ktorá priradí každému item-u vlastné identifikačné číslo a pomocou neho môžeme k item-u pristupovať. Ak sa v repeatry nachádza veľa záznamov je možné vytvoriť stránkovanie. Stránkovanie je definované pomocou parametra AllowPaginating. Ďalej si môžeme nastaviť veľkosť strany, tým je myslené koľko záznamov bude zobrazených na jednej strane. Potom už len asociujeme náš repeater s TPager pomocou parametra ControlToPaginate. Funkcie, ktoré som k jednotlivým značkám popísal nie sú zďaleka kompletné, slúžia len na objasnenie problému a demonštráciu príkladu.

Práca s technológiou AJAX

Ajax má veľmi silné zastúpenie v PRADO frameworku. Jeho čaro spočíva v možnosti asynchrónneho odosielania a prijímania údajov. Dnešné veľké internetové stránky sa vždy snažia docieliť vizuálny efekt pomocou ajaxu. Avšak nesmieme zabúdať nato, že AJAX je postavený na viacerých technológiách a jednu z nich je možné vo svojom prehliadači zakázať, konkrétne sa jedná o JavaScript. Preto je nutné navrhnúť funkčnosť stránok tak, aby nebola založená výhradne na JavaScripte, ale vždy existovala alternatíva. Požiadávky sú asynchronne odosielané na server a následne je spať odoslaný výsledok. Táto funkcionalita je možná bez akého-koľvek refreshu stránky. PRADO poskytuje širokú škálu komponentov využívajúcich ajax a sú rozpoznateľné od ostatných značiek tým, že začínajú slovom Active. Napríklad:

- <com:TButton> – predstavuje synchrónne odosielané tlačidlo
- <com:TActiveButton> – predstavuje tlačidlo využívajúce technológiu Ajax

Práca s transakciami

Transakcie sú veľmi dôležitou súčasťou pri aplikáciách, ktoré pracujú s databázami. Keďže aplikácia dovoľuje vykonávať viacerým užívateľom viacej operácií naraz (čítanie, zapisovanie), je dôležité zabezpečiť to, aby bola informácia predaná ďalej v atomickom, konzistentnom tvare a informácie ostanú nezávislé a v trvácnom tvare. Táto funkčnosť sa dá dosiahnuť použitím transakcií. Transakcie v PRADO framework reprezentuje inštancia triedy TDbTransaction. Práca s transakciami je znázornená a popísaná v nasledujúcom kóde:

1. Zahájenie začiatku transakcie.
2. Vykonávanie príkazov jeden po druhom. Pritom žiadne zmeny v databáze nie sú viditeľné navonok.
3. Ak prebehne vykonanie transakcie úspešne, zmeny sa stanú viditeľné aj navonok.
4. Ak, čo len jeden z vykonávaných dotazov zlyhá, je celá transakcia vrátená späť.

```
$members = new MembersRecord();
$members->DbConnection->Active = true;
// začiatok
$transaction = $members->DbConnection->beginTransaction();
try {
// vykonávané príkazy
$members = $members::finder()->findByPk($this->id->Text);
$members ->name = $this->name->Text;
$members ->surname = $this->surname->Text;
//... ďalšie príkazy
$members->save();
// úspešné vykonanie transakcie
$transaction->commit();
}
catch(Exception $e) {
// neúspešné vykonanie transakcie
$transaction->rollBack();
}
```

Fragment kódu použitia transakcii v PRADO framework.

Keď si vezmeme napríklad ekonomickú sféru (bankovníctvo, účtovníctvo) ale aj iné sféry tak využívanie transakcií je nevyhnutnosťou pre zachovanie korektných informácií, pretože na tom stojí celý informačný systém. Operácie nad databázou musia pracovať tak, aby nenastala žiadna kolízna alebo neočakávaná situácia.

Záver

Tento framework poskytuje veľa zaujímavých prínosov a je na každom z nás, či je ochotný učiť sa niečo nové, alebo bude račej pracovať na najnižšej úrovni a to úrovni PHP. Framework je určený na rýchly vývoj dynamických stránok. Hlavnou myšlienkou frameworku je uľahčiť prácu programátorov pri vyvíjaní webových aplikácií a znížiť potrebu základných znalostí programátorov na najnižšej vývojárskej úrovni. PRADO je predchodcom frameworku Yi a myslím si, že jeho funkcionalita pokrýva väčšinu potrieb aj náročných vývojárov. V prípade, že by vývojár nenašiel to čo hľadá, stále má možnosť vytvoriť si svoj vlastný komponent. Ďalším užitočným prínosom je oddelenie aplikačnej logiky od prezentačnej vrstvy. Toto rozdelenie podstatne zvyšuje prehľadnosť kódu ale čo je dôležitejšie, pri veľkých projektoch nám vznikne možnosť rozdelenia úloh do viacerých tímov. Jeden tím môže pracovať na vzhľade aplikácie, zatiaľ čo iný tím môže vytvárať funkcionalitu aplikácie.

Tvorcovia frameworku dbali na dodržiavanie bezpečnostných princípov a validity kódu podľa štandardov konzorcia W3C. Týmto krokom prispeli k tomu, že aplikácie vyvíjané v tomto frameworku sa zobrazia koncovému užívateľovi totožne v akomkoľvek internetovom prehliadači. Cache moduly zvyšujú rýchlosť stránok a znižujú výpočtovú náročnosť pri opakovanom prístupe ku stránkam. Šablóny nám slúžia na odstránenie redundantného kódu a prispievajú k zvýšeniu efektívnosti a prehľadnosti pri programovaní. Komunikácia s databázou je plne podporovaná v rátane využitia transakcií. Pre menšie projekty je určený skorej modul ActiveRecord a pre veľké zasa DataMapper v ktorom si môžeme sami vytvárať vlastné SQL dopyty. Taktiež sa

môžeme tešiť širokej škále komponentov využívajúcich technológiu AJAX ktorá vytvára estetický dojem na koncového užívateľa. Pri správnom použití AJAXU môžeme znížiť náročnosť na komunikáciu s databázou, avšak v nesprávnych rukách môže byť efekt záporný.

Všetky nastavenia aplikácie sú ukladané v súbore application.xml a prístupové pravidlá zasa v súbore config.xml. Vytvoriť novú aplikáciu môžeme jednoducho pomocou pár príkazov v príkazovom riadku (cmd) ktorý nám vytvorí celú kostru nového projektu. Tvorcovia frameworku (nie všetci) sa zamerali na vývoj nového frameworku Yi, ktorý by mal byť jeho nástupcom. Takže to spôsobilo, že na frameworku sa už nepracuje tak ako v minulosti a pohľad sa upiera skorej no nového nástupcu. Všetky potrebné informácie nájdete na domovskej adrese frameworku http://www.pradosoft.com. Ako jedinú nevýhodu by som videl nutnosť štúdia dokumentácie, pretože v opačnom prípade framework nemusí kladne pôsobiť na strávení čas pri vývoji aplikácií, ale naopak vývojár bude strácať čas s hľadaním vhodného riešenia.