Karol Przystalski

Symfony. Aplikacje internetowe w praktyce

PWN

Redakcja: **Joanna Cierkońska**

Skład komputerowy: **Krzysztof Świstak**

ISBN: 978-83-01-15726-5

Wydawnictwo Naukowe PWN SA
00-251 Warszawa, ul. Miodowa 10
tel. (0 22) 69 54 321
faks (0 22) 69 54 031
e-mail: pwn@pwn.com.pl
www.pwn.pl

Wydawnictwo Naukowe PWN SA
Wydanie pierwsze,
Arkuszy drukarskich 37,5
Druk ukończono w listopadzie 2008 r.
Druk i oprawa:
ZWP HEL,
04-007 Warszawa,
ul. Grenadierów 77

Spis treści

Podziękowania

Niniejsza książka jest pierwszą w moim dorobku. Napisanie jej było dla mnie ogromnym wyzwaniem. Dziękuję przede wszystkim pracownikom redakcji informatycznej Wydawnictwa Naukowego PWN. Bez ich pomocy nie dałbym rady. W szczególności podziękowania kieruję do Artura Nowotarskiego, który pomagał mi na początku pisania, do Magdy Ścibor oraz Iwony Krajewskiej-Kranas za cenne uwagi, które spowodowały, że zrewidowałem swoje poglądy odnośnie oprogramowania. Dziękuję Joannie Cierkońskiej za opracowanie redaktorskie – bez jej poprawek książka byłaby zapewne zrozumiała tylko dla nielicznych czytelników, oraz Krzysztofowi Świstakowi za skład i wprowadzenie wszystkich poprawek.

Dziękuję także Agacie przede wszystkim za cierpliwość.

Mamie

Wstęp

O czym jest ta książka?

W książce tej chciałbym pokazać, jak proste jest napisanie dobrej aplikacji webowej. Ma ona także nauczyć początkujących programistów dobrych nawyków pisania aplikacji z elementami zarządzania projektami.

Dla kogo jest ta książka?

Książka ta jest przeznaczona dla osób rozpoczynających pisanie aplikacji webowych oraz dla tych, którzy pisali już aplikacje w języku PHP lub innym, ale nie mają doświadczenia w tworzeniu aplikacji za pomocą jednego z frameworków. Na podstawie jednego z najlepszych obecnie frameworków postaram się pokazać, jak implementować wzorce projektowe, przedstawię podstawy refaktoryzacji kodu, model MVC oraz wiele innych technik i standardów używanych nie tylko w Symfony.

Dokumentacja Symfony

Dokumentacja w Symfony jest dobrze rozbudowana. Należy z niej korzystać, ponieważ jest jedną z najlepiej opracowanych wśród frameworków webowych. Ta książka jednak idzie o krok dalej. Przede wszystkim omawiany jest przypadek praktycznego zastosowania nie tylko pod kątem znajomości Symfony, ale także inżynierii oprogramowania, tak aby projekt był przydatny, skalowalny oraz czytelny w przyszłości.

Rozdział 1

Podstawy tworzenia aplikacji webowych

W ostatnich czasach panuje moda na serwisy społecznościowe, dlatego wybrałem taki serwis jako przykład aplikacji webowej, który posłuży do nauki. Wiemy już, co będziemy pisać, więc na początek należy zapoznać się z Symfony oraz kilkoma podstawowymi pojęciami dotyczącymi inżynierii oprogramowania.

1.1. Dlaczego Symfony?

Jest wiele innych frameworków, które są umożliwiają pisanie w językach Perl, Python, Ruby czy Smalltalk – przykładowe wymieniono w tabeli 1.1.

Tabela 1.1. Frameworki webowe wg języków programowania

Ruby	PHP	Python	Perl	Smalltalk	Java
Ruby on Rails	Symfony	Django	Catalyst	Seaside	Spring
Nitro	CakePHP	Turbogears	Maypole	-	Struts
Merb	CodeIgniter	Pylons	Jifty	-	Faces

Oczywiście serwis, który nas interesuje, można napisać za pomocą każdego z wyżej wymienionych frameworków – to tylko narzędzia, ale w jednych pisze się wygodniej, w drugich mniej. Wybór zależy też od upodobania np. do języka lub wymogów specyfikacji. Spośród powyższych Symfony najbardziej przypomina Ruby on Rails. W trakcie tworzenia Symfony programiści zapożyczyli z niego wiele pomysłów. Nawet nazewnictwo jest bardzo podobne, np. Pake i Rake. Stało się tak, ponieważ RoR był pierwszym znanym i cenionym frameworkiem webowym, dodatkowo opartym na wolnej licencji. Zdobył popularność dzięki innowacyjnym rozwiązaniom, które zostały w nim zastosowane. Programiści Symfony postarali się najlepsze z nich przenieść do swojego frameworku.

Trochę historii

Symfony oparto na języku PHP, z którym wiąże się stereotypowa opinia, że jest „dziurawy". Ta opinia funkcjonuje, niestety, nadal, a wiąże się z pewnymi zaszłościami. PHP, oparty kiedyś na skryptach w Perlu, później już samodzielnie zdobył ogromną popularność. Do dzisiaj jest językiem najczęściej używanym do pisania aplikacji webowych. Był dosyć prężnie wówczas rozwijany, co niestety powodowało wiele usterek w nowych aplikacjach, za których powstanie obwiniano błędy w języku. Nie byłoby w tym nic nadzwyczajnego, ponieważ każdy język, jak każdy program, zawiera błędy. Jednak PHP zdobył tak ogromną popularność, że zaczęli się go uczyć wszyscy. Prostota okazała się w tym wypadku zarówno dużym atutem, jak i pewnym problemem. Język zyskał popularność dzięki swojej prostocie, ale także ożliwościom, które wtedy dawał. Każdy początkujący programista dość szybko mógł się go nauczyć ze względu na łatwą do przyswojenia gramatykę. Można to porównać do języków C i Pascal. Pascal wymagał pewnego porządku: najpierw deklaracja zmiennych, później kod itp., natomiast w C można zadeklarować zmienną w środku kodu. (Dzięki takiemu nastawieniu C, a później C++ zdobyły rzeszę zwolenników. Oczywiście jest to jeden w wielu powodów, niemniej jednak nastawienie na wygodę programisty jest w tym wypadku jednym ze znaczących czynników). PHP również zawiera udogodnienia, które ułatwiają życie programiście. Dzięki temu powstało mnóstwo aplikacji, które zdominowały Internet, ale sporo z nich miało luki w zabezpieczeniach i tak powstała przywołana wyżej opinia. Inne zarzuty to:

- za wolny,
- zbyt prosty,
- nie można w nim napisać dobrej, profesjonalnej aplikacji.

Nieważne, jakich narzędzi programista używa, ważne, co potrafi za ich pomocą stworzyć. Uważam, że PHP jest świetny do tworzenia aplikacji webowych. Wielu jego przeciwników uważa, że jest za mało obiektowy. Nie ma to zasadniczego znaczenia dla tworzenia naszej aplikacji ani też nie umniejsza jego walorów. Należy pamiętać, że został utworzony jako język do pisania skryptów z przeznaczeniem do aplikacji webowych. Co innego można powiedzieć o Ruby, Pythonie czy Smalltalku. Oczywiście doskonale oddają ideę języka obiektowego i obecnie są poważną konkurencją dla PHP. Jednak trudno im będzie zyskać podobną popularność, bo choć na jego temat krążą stereotypowe opinie, to jednak zdobył sławę jako prosty i dobry język do wspomnianych rozwiązań.

Zalety Symfony

Ostatnimi czasy powstało wiele frameworków w języku PHP, w tym między innymi Symfony. Symfony jako framework można traktować tak jak inny, zupełnie

nowy język. Zachowuje całą gramatykę oraz funkcjonalność PHP, ale sposób pisania aplikacji jest bardzo odmienny od monolitycznego czy modułowego. Pomijam sposób monolityczny, ponieważ obecnie stosowanie go ma sens jedynie przy programowaniu ekstensywnym. Wady poprzedniego sposobu to m.in. mnóstwo plików konfiguracyjnych w różnych katalogach, brak stałych schematów, struktury plików. Wszystko było napisane dosyć chaotycznie, chociaż logicznie poprawnie. Problemy pojawiały się podczas pisania kodu, który powinien zostać użyty wiele razy. Zgodnie z zasadą DRY (ang. *don't repeat yourself*) niektóre partie można napisać łatwiej i tylko raz. Omówimy to w części poświęconej refaktoryzacji oraz MVC. W Symfony zastosowano model MVC, a sam framework wymusza pisanie czystego kodu i skalowalnych aplikacji. Oto najważniejsze z wielu zalet pisania aplikacji za pomocą Symfony:

- logiczna struktura katalogów,
- sporo możliwości konfiguracyjnych,
- CRUD,
- ORM,
- Admin generator,
- model MVC,
- domyślna obsługa AJAKSA,
- ogromna liczba dodatków oraz wtyczek,
- świetna dokumentacja,
- zabezpieczenie przed atakami typu injection itp.

Natomiast z punktu widzenia e-biznesu można wyróżnić również takie zalety:

- programiści PHP są tańsi i łatwiej dostępni niż np. programiści Ruby, Pythona czy Smalltalka, ze względu na popularność języka,
- ogromna większość firm oferujących hosting ma domyślnie włączoną obsługę skryptów napisanych w języku PHP,
- stosunkowo dużo niższy koszt niż rozwiązania oparte na .NET czy Javie,
- licencja MIT,
- szybkie tworzenie aplikacji dzięki połączeniu MVC z Agile.

Nietrudno znaleźć programistę znającego PHP. Troszkę trudniej znaleźć dobrego programistę PHP, ale to dużo łatwiejsze niż znalezienie programisty Ruby czy Pythona, o programistach Smalltalka nie wspominając. Konkurencja między programistami PHP jest na tyle duża, że średnio są tańsi niż pozostali. Jest to dosyć istotne podczas planowania projektu, zwłaszcza kosztów. Zaletą dla potencjalnego pracownika na stanowisku programisty PHP jest to, że nie ma poważnych problemów ze znalezieniem

pracy ze względu na dominację tego języka na rynku aplikacji webowych. Podobnie sytuacja ma się z hostingiem, o który raczej nie należy się martwić. Język PHP, tak jak i Symfony, oparty jest na otwartych, wolnych licencjach, co powoduje obniżenie kosztów. Ma to zaletę w stosunku do komercyjnych rozwiązań, takich jak ASP czy JSP, które w dodatku często potrzebują dużo więcej zasobów, a zatem zwiększają koszty projektu. Aby przekonać się o zapotrzebowaniu na programistów PHP, wystarczy wejść na strony jednego z wielu znanych serwisów ogłoszeniowych z ofertami pracy. W jego wyszukiwarce należy najpierw wpisać frazę: „Programista PHP", a później kolejno: „Programista Ruby" i „Programista Python". Ja znalazłem w pierwszym wypadku 51 ogłoszeń, a w pozostałych odpowiednio 2 i 6 ogłoszeń. Różnicę po prostu widać. Zaryzykuję twierdzenie, że nie zmieni się to w ciągu kolejnych kilku lat.

Nowości w Symfony 1.1

Symfony 1.1 zawiera wiele udoskonaleń w porównaniu do poprzedniej wersji, jak to zazwyczaj bywa. Do nowości możemy zaliczyć:

- refaktoryzację starej architektury – obecnie jest łatwiejszy dostęp do wszystkich elementów struktury Symfony;
- zastąpienie systemu zadań batch systemem tasks, dającym większe możliwości; zostały one zintegrowane z CLI (ang. *command line interface*) Symfony;
- nowy system tworzenia formularzy i walidacji – teraz odbywa się to bezpośrednio w kontrolerze, dzięki czemu usunięto helpery;
- nowy parser YAML – obecnie wyświetla zrozumiałe błędy podczas parsowania plików .yml;
- nowy system wtyczek, zgodny z PEAR – pozwala na zarządzanie wtyczkami w projekcie.

Dodatkowo istnieje możliwość zapewnienia kompatybilności wstecz z Symfony 1.0. Powoli widać tendencję do wycofywania się z ORM, jakim jest Propel, wbudowanym na stałe. Zmiany idą w tym kierunku, aby ORM był dodawany jako wtyczka, co umożliwia zainstalowanie nowego Propel 1.3 czy Doctrine.

Nowości w Symfony 1.2

Różnice między Symfony 1.1 a 1.2 są znacznie mniejsze niż między wersjami 1.1 i 1.0. Główne rozbieżności to wynik refaktoryzacji kodu, która umożliwiła wprowadzenie kolejnych funkcji. Jedyne części znacznie zmodyfikowane to admin generator oraz routing. Główną przyczyną zmiany generatora jest dostosowanie go do nowego systemu tworzenia formularzy, wprowadzonego w Symfony 1.1. Zmiany w wersji 1.2. pokazują kierunek rozwoju Symfony. W 1.2. moduł do obsługi bazy da-

nych to – do wyboru – Doctrine albo Propel. Różnica między obiema wersjami jest jednak taka, że w 1.2. nie ma domyślnego modułu obsługi bazy danych. Oba moduły zostały „wyrzucone" do osobnych wtyczek. W taki sposób twórcy nie narzucają jednego rozwiązania. Podobnie jest z modułami JavaScriptu do obsługi technologii AJAX – również do wyboru jest kilka frameworków w formie wtyczek. Pozostałe zmiany to:

- możliwość personalizacji panelu debugowania (panel w prawym górnym rogu), który obecnie jest inspiracją dla innych frameworków (np. Django);
- dodanie biblioteki Swift jako domyślnej do obsługi e-maili (korzystamy z niej w książce jako wtyczki do Symfony 1.1.).

Wróćmy jednak do routingu: główną zmianą jest możliwość dodania innych znaczników oddzielających poszczególne pola, np. zamiast / można dać – albo \ czy kropkę. Nowością jest także wprowadzenie obsługi REST (ang. *representational state transfer*), czyli takiej alternatywy dla RPC (ang. *remote procedure call*), która wykorzystuje metody wywołania protokołu HTTP: POST, GET, PUT, DELETE. PUT i GET to te same, którymi wysyłamy formularze. Dzięki wykorzystaniu wszystkich metod można napisać aplikację, która działa jak moduł CRUD, korzystając jedynie z metod protokołu HTTP.

1.2. Jak zacząć?

Przyznam, że jest to rozdział, którego nie lubię, ponieważ opisuje rzeczy, które trzeba zrobić, a zostały już wszędzie opisane. Aby zacząć przygodę z Symfony, należy najpierw rozważyć problem zależności.

1.2.1. Zależności

Aby Symfony zostało poprawnie zainstalowane, należy uwzględnić również takie pakiety, jak:

- Phing,
- Pake,
- Lime,
- Propel,
- Creole.

PEAR

W systemach z rodziny Windows należy najpierw zainstalować WAMP (Windows, Apache, MySQL, PHP), a dopiero później PEAR. Opisano to w podrozdziale poświęconym instalacji w Windows. W systemach Mac OS X należy ściągnąć PEAR i dodać do systemu:

```
# curl http://pear.php.net/go-pear > go-pear.php
# sudo php -q go-pear.php
```

W systemach Linux w zależności od dystrybucji należy posłużyć się danym menedżerem pakietów. W zależności od dystrybucji przez menedżer pakietów instalujemy:

- Gentoo – `emerge PEAR-PEAR`,
- Debian i pochodne, np. Ubuntu – `apt-get install php-pear`,
- Suse – `yum php-pear`,
- Red Hat i pochodne – `rpm php-pear*.rpm`.

Najczęściej używanymi opcjami PEAR będą *channel-discover* oraz *install*, odpowiadające za utworzenie listy pakietów danego kanału, przez który pakiet będzie ściągany. Install, jak sama nazwa wskazuje, instaluje pakiet, ale informacje o istnieniu tego pakietu muszą zostać wcześniej ściągnięte z kanału.

Phing

Phing jest pakietem używanym przede wszystkim podczas tworzenia modelu za pomocą narzędzia Propel. Aby go zainstalować, należy najpierw dodać kanał w PEAR, a później zainstalować pakiet przez PEAR:

```
# pear channel-discover pear.phing.info
# pear install phing/phing
```

Pake

Pake jest odpowiedzialny za scaffolding (z ang. rusztowanie) oraz CRUD. Generuje pliki dla aplikacji oraz modułów, a także jest odpowiedzialny za wszelkie generatory w projekcie. Instalujemy go następująco:

```
# pear channel-discover pear.symfony-project.com
# pear install symfony/pake
```

Lime

Lime jest odpowiedzialny za testy funkcjonalne oraz jednostkowe, rozwijane przez programistów Symfony, i dostępny *out of the box*.

Propel

Propel generuje model oraz klasy, jest domyślnym ORM w Symfony. Instalacja:

```
# pear channel-discover pear.phpdb.org
# pear install phpdb/propel_generator
# pear install phpdb/propel_runtime
```

Propel można także zainstalować z poziomu Symfony za pomocą (dotyczy Symfony w wersji 1.1 i starszej):

```
$ Symfony plugin:install sfPropelPlugin
```

Instalacja Propela jest potrzebna tylko w Symfony 1.2. W wersji 1.1. można natomiast zastąpić biblioteki nowszymi, z Propela 1.3, który wprowadza wiele nowości i jest przede wszystkim dużo szybszy.

Creole

Creole jest warstwą abstrakcyjną bazy danych. Pozwala na łączenie z wieloma bazami danych. Instalacja:

```
# pear channel-discover pear.phpdb.org
# pear install phpdb/creole
```

1.2.2. Instalacja w systemach Unix (Linux, Mac OS)

Najpierw należy Symfony zainstalować w systemie. Można to zrobić co najmniej na wymienione poniżej cztery sposoby:

- ściągnięcie źródeł i rozpakowanie do żądanego katalogu,
- PEAR,
- SVN,
- systemowy menedżer pakietów.