

Tworzenia aplikacji webowych z Zope X3

FORMUŁA X

Zope, serwer aplikacji sieciowych napisany w języku Python, to wyjątkowo popularna platforma systemowa do zarządzania treścią, funkcjonująca na zasadzie Open Source. Niedawno ukazała się jej nowo utworzona wersja – X3.0. Przedstawiamy najważniejsze nowe cechy i funkcje Zope X3.0.

PHILIPP VON WEITERSHAUSEN

Od momentu pojawienia się systemu zarządzania treścią (ang. content management system – CMS) Plone [1], trudno sobie wyobrazić świat wolnych serwerów aplikacji sieciowych bez Zope [2]. Powodzenie tego projektu dowodzi, że Zope z łatwością utrzymuje przewagę nad innymi produktami tego typu, opartymi na platformie Java. Język programowania Python wnosi do świata Zope obiektowy, modułarny kod źródłowy, a także elastyczność i łatwość wprowadzania zmian, kojarzone częściej z sieciowymi językami skryptowymi.

Od CMF do zupełnie nowych konstrukcji

Zope jest popularną platformą CMS. Większość wolnych systemów CMS działających w środowisku Zope (np. Plone [1], Silva [3] i CPS [4]) wykorzystuje produkt zwany Content Management Framework (CMF) – platformę zarządzania treścią. Platforma CMF dla Zope 2 wyposażona jest w narzędzia niezbędne do tworzenia systemu zarządzania treścią, a jej komponentowa architektura umożliwia elastyczne modyfikowanie poszczególnych komponentów. Podczas tworzenia środowiska Zope wykorzystano doświadczenia zdobyte przy tworzeniu platformy CMF.

Trzy lata temu Zope Corporation zdecydowała się nie przerabiać skomplikowanego kodu źródłowego Zope, ale napisać go od nowa. Ze względu na mocne zaangażowanie w kwestie Wolnego Oprogramowania, Zope miał zawsze silne poparcie wśród społeczności informatycznej. Aby utrzymać tempo rozwoju projektu, doświadczeni i początkujący programiści Zope organizowali na całym świecie główne i pomniejsze sesje programistyczne (tzw. sprinty). Jakość projektu zapewniono za pomocą takich metod jak programowanie

ekstremalne (ang. Extreme Programming – XP) i testowanie modułów. Ponadto fakt, że wersja Zope oznaczona numerem 2 jest stabilna i cieszy się powodzeniem, dał twórcom nowej wersji dość czasu na zoptymalizowanie kluczowych funkcji w powtarzających się krokach. Nikt także nie narzekał na konieczność pozbycia się fragmentu kodu, aby zrobić miejsce dla lepszych implementacji.

Charakterystyka

Rdzeniem Zope X3 jest jego komponentowa architektura. Różne komponenty odpowiadają za konkretne zadania – przechowywanie, przetwarzanie i prezentację danych.

Wielu Czytelników zna zapewne bazę ZODB (Zope Object Database) z Zope 2. Obiekty w bazie ZODB mogą istnieć w sposób całkowicie przezroczysty. Baza jednocześnie wyposażona jest w wykorzystywane w przedsiębiorstwach funkcje, takie jak śledzenie i kontrola transakcji oraz interfejs wewnętrznej pamięci. Moduł ZEO (Zope Enterprise Objects) obsługuje nawet klastry wielu instancji ZOPE, co znacznie ułatwia skalowalność.

Elastyczny system bezpieczeństwa przydziela uprawnienia w celu ochrony komponentów, atrybutów i metod. Użytkownicy, którzy chcą mieć dostęp do komponentów chronionych, muszą uzyskać autoryzację. Modułowość komponentów, odpowiadających za uwierzytelnianie użytkowników i nadawanie im uprawnień, daje operatorom możliwość dostosowywania poziomu bezpieczeństwa bez konieczności przerabiania całej aplikacji.

System Zope nie tylko wyposażony jest w narzędzia do lokalizacji, niezbędne do obsługi różnych języków – jest w pełni międzynarodowy. Pełna paleta funkcji Zope obejmuje usługi pocztowe, oparte na protokole

SMTP albo na programie Sendmail, system powiadamiania o zdarzeniach i obsługa XML RPC.

Zmiana paradygmatów

W Zope X3 usunięto niektóre z problemów, dotyczących Zope 2. We wcześniejszej wersji od instancji klas oczekiwano na przykład podawania atrybutów i metod, koniecznych do współpracy z Zope. Programiści przeładowywali więc obiekty wieloma różnymi metodami, co ograniczało przenośność i utrudniało późniejsze wprowadzanie zmian w funkcjach.

W Zope X3 poszczególne komponenty są natomiast jak najprostsze, a dodawanie nowych funkcji odbywa się za pomocą dodawania nowych komponentów. W skład architektury Zope wchodzi następujące typy komponentów:

- Komponenty typu *Content* (treść) nie mają zwykle metod, tylko własności, wykorzystywane do publikowania przechowywanych danych. Typy i wartości danych określone są zwykle za pomocą schematów danych.

Listing 1: Interfejsy (*interfaces.py*)

```
from zope.interface import z
Interface
from zope.schema import Text, z
TextLine
class IBuddy (Interface):
    """Information about a buddy"""
    first = TextLine(title=u"firstname")
    last = TextLine(title=u"familyname")
    address = Text(title=u"address")
    zipcode = TextLine(title=u"zipcode")
```

- Komponenty typu *Utilities* (usługi) to składniki niezależne od kontekstu, odpowiedzialne za wykonywanie konkretnych zadań, np. łączenie się z bazą danych, indeksowanie czy dostarczanie poczty elektronicznej.
- Komponenty typu *Adapters* (adaptery) mają największe możliwości. Umożliwiają dodawanie funkcji do istniejących komponentów bez konieczności ich modyfikowania. Technika ta jest szczególnie przydatna wtedy, kiedy dana platforma wymaga wykorzystania określonego API. Programiści nie muszą zmieniać oryginalnego komponentu – wystarczy, że zaimplementują adapter, pełniący funkcję interfejsu między danym komponentem a wymaganym API.
- Komponenty typu *Views* (widoki) pozwalają na graficzne zaprezentowanie innych komponentów użytkownikowi. Przykładem jest przeglądarka WWW, wykorzystująca „widoki” wyświetlające HTML. Widok jest właściwie specjalnym rodzajem adaptera, przydzielającego innym komponentom funkcję prezentacji, której same z siebie nie mają.

Abstrakcyjny układ

Aby zapewnić niezależność komponentów od implementacji, nie odwołujemy się do nich za pomocą klas. Do opisanego zestawu funkcji danego komponentu wykorzystujemy natomiast interfejsy. Są one czymś w rodzaju „formalnej umowy”, gwarantującej udostępnienie konkretnej funkcji w formie narzuconej przez API. Ponieważ język Python nie używa interfejsów, zostały one wynalezione na potrzeby Zope.

Listing 2: Content Components (*buddy.py*)

```
from persistent import Persistent
from zope.interface import z
implements
from buddydemo.interfaces import IBuddy
class Buddy(Persistent):
    implements(IBuddy)
    def __init__(self, first='', z
last='',
        address='', zip_code=''):
        self.first = first
        self.last = last
        self.address = address
        self.postal_code = z
        postal_code
```

Na Listingu 1 przedstawiono interfejs pochodzący z przykładowej aplikacji służącej do zarządzania adresami. Zaprezentowany interfejs to schemat danych, opisujący komponenty typu Content – w tym przypadku dane adresowe znajomego („buddy”). Przechowywanie kodu pocztowego pozwala na późniejsze wyszukiwanie informacji o mieście i stanie (województwie).

Ponieważ Python nie ma wbudowanych interfejsów, do zdefiniowania interfejsu na Listingu 1 służy słowo kluczowe języka Python – *class*. W Zope nie ma ponadto rozróżnienia między interfejsami opisującymi swoje działanie za pomocą metod a interfejsami definiującymi schematy danych.

Proste komponenty typu Content

Napisanie trwałej (nieulotnej) klasy w Zope 2 było dość skomplikowanym zadaniem. Do stworzenia nowych instancji, wymaganych przez interfejs sieciowy, potrzebowaliśmy co najmniej metatypu, deklaracji bezpieczeństwa i metod tworzących te instancje. Na szczęście, wymagania te zostały teraz zmniejszone do minimum. Trwałe obiekty muszą tylko obsługiwać przesyłane do nich dane; za obsługę całej reszty odpowiadają inne komponenty.

Na Listingu 2 zaprezentowana jest działająca implementacja interfejsów *IBuddy* z Listingu 1. Warto zauważyć, że klasa dziedziczy po klasie *Persistent* – dzięki temu jej wystąpienia są automatycznie zapisywane w ZODB. Aby zapewnić zgodność tej klasy z innymi komponentami, trzeba także określić, że implementuje ona interfejs *IBuddy*.

Konfiguracja oparta na XML

W Zope 2 biblioteki importowane były z katalogu *Products*. Za rejestrację komponentów odpowiadał moduł inicjujący każdego pakietu (*__init__.py*). Inne elementy, takie jak deklaracje bezpieczeństwa lub konfiguracje widoku przeglądarki, znajdowały się w kodzie aplikacji.

W Zope X3 zastosowano inne podejście. Na przykład rozszerzenia Zope to teraz proste pakiety napisane w języku Python – oznacza to, że można je zainstalować w dowolnym miejscu, jeśli tylko znajdują się w zmiennej środowiskowej *PYTHON-PATH*. Wszystkie inne elementy związane z konfiguracją komponentów – sama rejestracja, deklaracje bezpieczeństwa czy wi-

Listing 3: Konfiguracja (*configure.zcml*)

```
<configure
    xmlns="http://namespaces.z
zope.org/zope"
    xmlns:browser="http://namespa-
ces.zope.org/browser">
<content class="buddydemo.z
buddy.Buddy">
    <require
        permission="zope.View"
        interface="buddydemo.z
interfaces.IBuddy" />
    <require
        permission="zope.z
ManageContent"
        set_schema="buddydemo.z
interfaces.IBuddy" />
</content>
<browser:addform
    schema="buddydemo.z
interfaces.IBuddy"
    label="Add a new buddyz
address"
    content_factory="buddydemo.z
buddy.Buddy"
    arguments="first lastz
address zipcode"
    name="AddBuddy.html"
    permission="zope.z
ManageContent" />
<browser:editform
    schema="buddydemo.z
interfaces.IBuddy"
    label="Edit buddy address"
    name="edit.html"
    menu="zmi_views" title="Edit"
    permission="zope.z
ManageContent" />
<browser:addMenuItem
    class="buddydemo.buddy.Buddy"
    title="Buddy"
    permission="zope.z
ManageContent"
    view="AddBuddy.html" />
</configure>
```

doki przeglądarki – przeniesione zostały do pliku konfiguracyjnego. Takie podejście daje programistom istotną korzyść: możliwość wyłączenia komponentów czasowo lub na stałe bez konieczności wprowadzania zmian w kodzie.

Na Listingu 3 zaprezentowane są typowe dyrektywy konfiguracyjne dla komponentu typu Content, opartego na schemacie; przy-



Rysunek 1: Formularz edycji, stworzony automatycznie przez odwołanie się do schematu danych komponentu typu Content.

kład obsługuje jedynie deklaracje bezpieczeństwa dotyczące zapisu i odczytu danych dla instancji klasy Buddy. Następnie przechodzimy do definiowania dwóch formularzy – jeden z nich tworzy obiekty Buddy, drugi służy do ich edytowania.

Zope ma możliwość automatycznego tworzenia formularza na podstawie schematu danych, zdefiniowanego w interfejsie *IBuddy*. Jak zatem widać na Rysunku 1, schemat, pro-

sta, trwała implementacja, kilka dyrektyw konfiguracyjnych i zero HTML tworzą razem komponent, działający w przeglądarce.

Ostatnia dyrektywa na Listing 3 odpowiada za dodanie do menu interfejsu sieciowego Zope wpisu, umożliwiającego tworzenie nowych rekordów dotyczących osób. Fakt, że dyrektywa ta w ogóle istnieje, ilustruje ważną podstawową zasadę filozofii Zope 3: „Lepiej wyraźnie niż w domyśle”

Instalacja i konfiguracja

Zainstalowanie Zope pod Linuxem jest proste. Potrzebujemy aktualnej wersji języka Python – 2.3.4 z obsługą Zlib. Zope napisany jest w większości w Pythonie, ale niektóre moduły, ze względu na szybkość, zaimplementowane zostały w języku C. Przed instalacją musimy skompilować tę aplikację. W archiwum tar.gz znajduje się skrypt *configure*, który automatycznie tworzy plik *Makefile*, wykorzystywany w procesach kompilowania i instalacji.

Biblioteki Zope są zwykle instalowane w katalogu */usr/local/Zope-3.0.x*. Możemy oczywiście zmienić ich lokalizację, zmieniając parametr *-prefix* w skrypcie *configure*. Aby uruchomić instancję serwera, musimy najpierw stworzyć dla niego drzewo katalogów. Posłuży ono nie tylko do przechowywania obiektowej bazy danych instancji Zope (ZODB), ale także wymaganych przez nią bibliotek pomocniczych. Pojedyncza instalacja Zope może oczywiście używać wielu równoległych instancji.

Skrypt *mkzopeinstance*, znajdujący się w katalogu *bin*, tworzy katalog instancji. Jeśli określimy ścieżkę dla instancji i poświadczenia (ang. credentials) dla tymczasowego konta administracyjnego, możemy daną instancję uruchomić, wpisując polecenie *runzope* (w katalogu *bin* danej instancji). Konfiguracja serwera, porty HTTP i FTP i różne opcje dotyczące logowania znajdują się w pliku */etc/zope.conf*. Format pliku konfiguracyjnego jest taki sam, jak format pliku konfiguracyjnego serwera Apache. Domyślnie instancja serwera HTTP korzysta z portu 8080, a serwer FTP – z portu 8021.

Po uruchomieniu Zope poleceniem *runzope* możemy nacisnąć klawisz Ctrl-C, aby opuścić program. Ponieważ w przypadku aplikacji serwerowej nie jest to najpraktyczniejsze rozwiązanie, możemy także uruchomić skrypt *zopectl* (również w katalogu *bin* danej instancji) – podobnie działa komenda *apachectl* serwera Apache.

Iksy i my

Krótko po rozpoczęciu prac nad Zope 3 stało się jasne, że konieczne będzie zrezygnowanie ze zgodności API z Zope 2. Symbol „X” przed numerem wersji miał początkowo oznaczać „eksperymentalność” projektu. Oczywiście stabilna wersja Zope X3.0 nie ma już nic wspólnego z eksperymentami, „X” ma jednak ostrzegać użytkowników, że Zope X3 nie jest zgodne ze swoją poprzednią wersją.

(ang. Explicit is better than implicit). Tworzenie oprogramowania w Zope 3 może wprawdzie oznaczać więcej pisania, ale przynajmniej odczytanie kodu za pół roku okaże się łatwiejsze.

Przyszłość

Od pewnego czasu Zope zyskuje coraz większą popularność – dzieje się tak ze względu na jego funkcje, elastyczność i fakt, że jest on oparty na języku Python. Wygląda na to, że społeczność zwolenników Zope będzie się nadal rozrastać. Wykorzystanie tego systemu w projektach realizowanych na dużą skalę i w dużych firmach pomogło mu dojrzeć. Zope X3 jest ogromnym krokiem naprzód w dziedzinie realizacji projektu. Rozpowszechnienie Zope X3 i jego nowego paradygmatu może potrwać, spodziewamy się więc, że przez pewien czas wykorzystywany będzie jeszcze Zope 2. Podziękowania należą się projektowi Five [6], pomagającemu w stopniowej i rozważnej migracji na Zope X3. ■

INFO

- [1] Plone: <http://plone.org>
- [2] Strona WWW społeczności Zope: <http://zope.org>
- [3] Silva: <http://infrac.com/products/silva>
- [4] CPS: <http://www.nuxeo.org/cps>
- [5] Zope X3 do pobrania: <http://zope.org/Products/ZopeX3>
- [6] Five: <http://codespeak.net/z3/five>

AUTOR

Philipp von Weikershausen studiuje fizykę w Dreźnie. Jest niezależnym programistą, konsultantem i członkiem zespołu twórców Zope 3. Wiosną 2005 r. ukaże się jego książka, *Web Component Development with Zope 3*.