Desenvolvendo aplicativos para a web com o Zope X3

Fórmula X

O programa de código aberto Zope, um servidor de aplicações web escrito em Python, é uma plataforma de gerenciamento de conteúdo extremamente popular. A versão X3.0, recentemente desenvolvida, acaba de ser lançada. O que será que há de novo?

Por Philipp von Weitershausen

Desde o lançamento do sistema de gerenciamento de conteúdo (ou CMS, *Content Management System*) *Plone* [1], é difícil imaginar um mundo de servidores livres de aplicações web sem o *Zope* [2]. O sucesso do projeto vem mostrar que o Zope pode facilmente ganhar terreno na batalha contra o Java, mesmo estando no campo desse último. A linguagem de programação Python traz um código fonte modular e orientado a objetos ao mundo do Zope, junto com a agilidade e a flexibilidade associadas mais tipicamente a linguagens de script web.

Do CMF a novos desenvolvimentos

O Zope é uma popular plataforma de CMS. A maioria dos sistemas CMS livres baseados no Zope (como Plone [1], *Silva* [3] e CPS [4]) usam um produto do Zope conhecido como Estrutura de Gerenciamento de Conteúdo (*Content Management Framework* – CMF). A estrutura CMF do Zope 2 não apenas tinha as ferramentas necessárias para desenvolver um sistema de gerenciamento de conteúdo, como também oferecia uma arquitetura baseada em componentes que permitia a modificação flexível de componentes individuais. A experiência ganha no desenvolvimento do CMF foi incorporada ao processo de desenvolvimento do Zope.

Em vez de retrabalhar o labiríntico código fonte do Zope, a Zope Corporation optou por começar do zero há três anos. Por causa de seu firme compromisso com o princípio do Software Livre, o Zope sempre teve o apoio de uma forte comunidade. Programadores experientes e novatos do Zope organizaram sessões de desenvolvimento grandes e pequenas (chamadas *sprints*, corridas de explosão) por todo o mundo para manter o projeto em movimento. Métodos como *Xtreme Programming* (XP) e o teste de unidades (*Unit Testing*) trouxeram uma garantia de qualidade ao projeto. Além disso, o fato de o Zope versão 2 ser um produto estável e bem-sucedido deu aos desenvolvedores tempo bastante para otimizar recursos chave em múltiplos passos de iteração. E ninguém se queixou por ter sido preciso cavar tantos fossos no código para abrir caminho a melhores implementações.

Quadro 1: Entendendo o X

Quando o trabalho de desenvolvimento do Zope 3 foi iniciado, os projetistas rapidamente perceberam que teriam que abandonar qualquer esperança de ter uma API compatível com a do Zope 2. O prefixo "X" nas versões antigamente indicava que ela era "experimental". Obviamente, a versão estável X3.0 do Zope é qualquer coisa, menos instável. O "X", entretanto, continuou no nome para deixar claro que o X3 não é compatível com os anteriores.

Recursos

O núcleo do Zope X3 é a arquitetura de seus componentes. Diferentes componentes receberam a responsabilidade por tarefas específicas, como armazenamento de dados, processamento e apresentação.

Muitos leitores devem ter familiaridade com o ZODB (*Zope Object Database*, Banco de dados de objetos do Zope) por causa do Zope 2. O ZODB permite que os objetos permaneçam bastante transparentes no banco de dados e, ao mesmo tempo, oferece recursos empresariais como rastreamento de transações, auditoria e uma interface para o subsistema de armazenamento. O ZEO (*Zope Enterprise Objects*, objetos empresariais do Zope) permite mesmo o agrupamento de diversas instâncias do Zope, o que facilita a escalabilidade.

Um sistema de segurança flexível atribui permissões para proteger componentes, propriedades e métodos. Os usuários que queiram acessar componentes protegidos precisam de autorização. Componentes modulares autenticam e autorizam usuários, dando aos operadores a capacidade de adaptar a postura de segurança sem ter que refazer o aplicativo.

O Zope não só tem as ferramentas de localização necessárias aos desenvolvedores para permitir a internacionalização de aplicativos: o próprio Zope é plenamente internacional. Uma rica gama de recursos, como SMTP ou serviços de email baseados no *sendmail*, um sistema de notificação de eventos e o suporte a XML RPC vêm completar a paleta do Zope.

Mudança de paradigmas

O Zope X3 não tem alguns dos problemas que afetavam o Zope 2. Por exemplo, a versão anterior esperava que a instanciação de classes fornecesse os atributos e métodos necessários para interagir com o Zope. Isso fazia com que os desenvolvedores precisassem sobrecarregar os objetos com uma grande variedade de métodos, o que limitava a portabilidade e dificultava muito mudanças subseqüentes nas funções.

O Zope X3, por outro lado, mantém os componentes individuais tão simples quanto possível e adiciona componentes se for necessário adicionar funcionalidade. A arquitetura do Zope engloba vários tipos de componentes:

- Os componentes de *Content* (*Conteúdo*) tipicamente não têm métodos, possuem apenas propriedades que usam para publicar os dados armazenados. Tipos e valores de dados são especificados normalmente com um esquema de dados.
- *Utilities* (Utilitários) são componentes independentes de contexto que realizam uma tarefa específica, como conexões a bancos de dados, indexação ou entrega de email.
- *Adapters* (Adaptadores) são provavelmente os componentes mais poderosos. Eles permitem que os desenvolvedores adicionem funcionalidade a componentes existentes sem a necessidade de modificá-los. Essa técnica é extremamente útil, já que uma estrutura requer uma API específica. Os desenvolvedores podem deixar o componente original como está e implementar um adaptador que faça a interface entre o componente e a API necessária.
- As *Views* (Visualizações) mostram os outros componentes para o usuário. Um navegador web seria um bom exemplo, pois usa visualizações que renderizam páginas HTML. Uma visualização é na verdade um tipo especial de adaptador que fornece aos outros objetos um recurso que eles não possuem nativamente (apresentação).

Listagem 1: Interfaces (interfaces.py)

```
01 from zope.interface import Interface
02 from zope.schema import Text, TextLine
03
04 class IBuddy(Interface):
05     """Informacao sobre um coleguinha"""
06
07     first = TextLine(title=u"nome")
08     last = TextLine(title=u"sobrenome")
09     address = Text(title=u"endereco")
10     zipcode = TextLine(title=u"CEP")
```

Contrato abstrato

Para permitir que os componentes permaneçam independentes da implementação, eles não são organizados por classe. Em vez disso, as interfaces são usadas para descrever a funcionalidade oferecida por um componente. Interfaces são um tipo de contrato formal que garante a provisão de uma função específica na forma de uma API. Como o Python não pode usar interfaces, o Zope as implementou ele mesmo.

A **listagem 1** mostra uma interface de um aplicativo de exemplo que gerencia endereços particulares. A interface mostrada aqui é um esquema de dados que descreve os componentes do conteúdo – neste caso, os dados do endereço de um amigo. A armazenagem do CEP permite buscar automaticamente as informações de cidade e estado mais tarde. ➠

Listagem 2: Componentes de conteúdo (buddy.py)

```
01 from persistent import Persistent
02 from zope.interface import implements
03 from buddydemo.interfaces import IBuddy
04
05 class Buddy(Persistent):
06     implements(IBuddy)
07
08     def __init__(self, first='', last='',
09                  address='', zip_code=''):
10         self.first = first
11    self.last = last
12    self.address = address
13    self.postal_code = postal_code
```

Listagem 3: Configuração (configure.zcml)

```
01 <configure
02     xmlns="http://namespaces.zope.org/zope"
03     xmlns:browser="http://namespaces.zope.org/browser">
04
05 <content class="buddydemo.buddy.Buddy">
06    <require
07        permission="zope.View"
08        interface="buddydemo.interfaces.IBuddy" />
09    <require
10        permission="zope.ManageContent"
11        set_schema="buddydemo.interfaces.IBuddy" />
12 </content>
13
14 <browser:addform
15     schema="buddydemo.interfaces.IBuddy"
16     label="Inserir o endere;o de um camaradinha"
17     content_factory="buddydemo.buddy.Buddy"
18     arguments="first last address zipcode"
19     name="AddBuddy.html"
20     permission="zope.ManageContent" />
21
22 <browser:editform
23     schema="buddydemo.interfaces.IBuddy"
24     label="Editar o endereco do companheiro"
25     name="edit.html"
26     menu="zmi_views" title="Edit"
27     permission="zope.ManageContent" />
28
29 <browser:addMenuItem
30     class="buddydemo.buddy.Buddy"
31     title="Buddy"
32     permission="zope.ManageContent"
33    view="AddBuddy.html" />
34
35 </configure>
```

Na **listagem 1**, a declaração `class` do Python é usada para definir uma interface, já que o Python não tem interfaces nativamente. Além disso, o Zope não distingue entre uma interface que usa métodos para descrever funcionalidade e uma interface que define um esquema de dados.

Componentes simples de conteúdo

A tarefa de escrever uma classe persistente no Zope 2 era bastante complexa. No mínimo, eram precisos o metatipo, declarações de segurança e métodos de instanciação para gerar as novas instâncias requeridas pela interface Web. Você ficará satisfeito em saber que agora as exigências são mínimas. Objetos persistentes só precisam lidar com os dados passados a eles; tudo o mais é manejado pelos outros componentes.

A **listagem 2** mostra uma implementação funcional das interfaces `IBuddy` mostradas na **listagem 1**. Note que a classe herda de `Persistent` para assegurar que suas instâncias sejam automaticamente armazenadas no ZODB. Para que essa classe se sobreponha sobre os outros componentes, também é preciso especificar que a classe implementa a interface `IBuddy`.

Configuração em XML

O Zope 2 precisava importar bibliotecas do diretório `Products`. O módulo de inicialização para cada pacote (`__init__.py`) costumava conter o registro do componente. Outros elementos, como declarações de segurança ou configurações de visualização do navegador, precisavam ser definidos e implementados no código do aplicativo final.

O Zope X3 adota uma abordagem diferente. Por exemplo, as extensões do Zope agora são apenas pacotes simples em Python, o que significa que você pode instalá-los quando bem lhe aprouver – desde que estejam no `PYTHONPATH`. Tudo o mais que seja relacionado com configuração de componentes, como o próprio registro, declarações de segurança ou visualização do navegador, agora está contido no arquivo de configuração. Esse método dá aos desenvolvedores uma importante vantagem: a capacidade de desabilitar componentes temporariamente ou permanentemente sem modificar o código.

A **listagem 3** mostra as diretivas típicas de configuração para um componente de conteúdo baseado em *esquemas* (*schema*); o exemplo mostra apenas as declarações de segurança para os dados de leitura e escrita para as instâncias de objetos "buddy" – ou seja, cada um dos seus amigos. O arquivo então passa a definir dois formulários: um deles gera objetos "buddy", o outro é usado para editá-los.

O Zope tem a capacidade de gerar automaticamente um formulário para o esquema de dados definido na interface `IBuddy`. Assim – como mostrado na **figura 1** – um esquema, uma implementação persistente simples, algumas diretivas de configuração e nenhum código em HTML dão forma a um componente que roda num navegador.

A última diretiva na **listagem 3** adiciona um item ao menu da interface web do Zope, item esse que permite aos usuários criar novos amiguinhos. O fato de essa diretiva existir exemplifica um importante princípio básico da filosofia do Zope 3: "Explícito é melhor que implícito". Embora o desenvolvimento de software com o Zope 3 possa significar mais digitação, ao menos será mais fácil ler o código depois de passados seis meses – principalmente se o desenvolvedor que o escreveu já não estiver mais disponível.

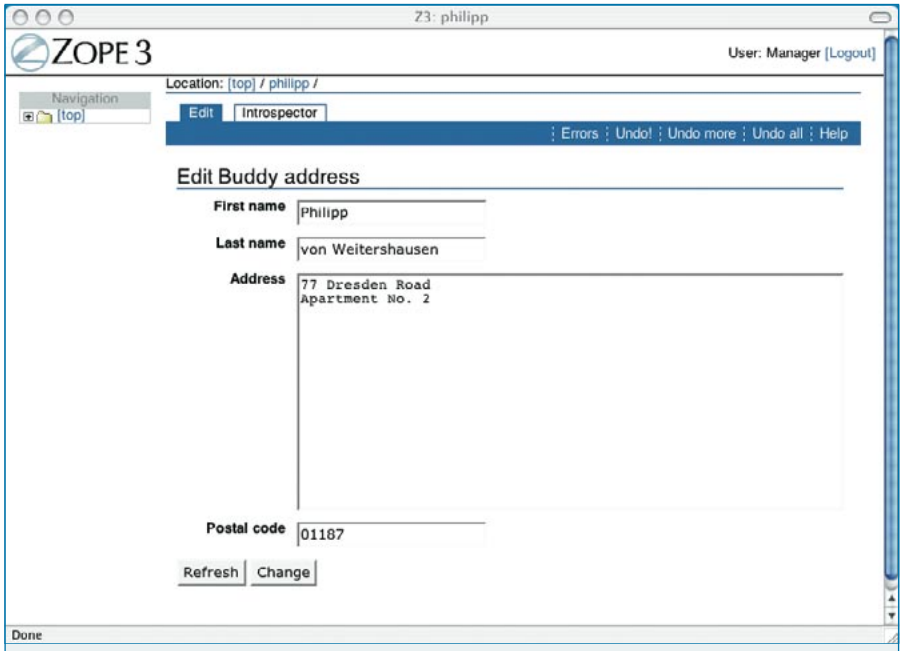

Figura 1: Esse formulário de edição foi gerado automaticamente por referência ao esquema de dados para o componente do conteúdo.

Um olhar sobre o futuro

O Zope tem milhões de seguidores fanáticos – tudo por causa de suas características, sua flexibilidade e sua dependência de algo tão popular quanto o Python. Ao que parece, essa comunidade só tende a crescer. O uso do software para projetos em larga escala e em grandes empresas ajudou o programa a alcançar a maturidade que vemos hoje. O Zope X3 é um passo de gigante dado à frente pelo pessoal que o mantém. Por outro lado, vai demorar um bocado até que o Zope X3 e seu novo paradigma ganhe as ruas, portanto devemos esperar ainda vida longa ao Zope 2. Graças ao projeto Five [6], podemos tentar migrar do Zope 2 para o X3 sem pensar muito sobre as incompatibilidades da API. ■

Quadro 2: Instalação e configuração

Instalar o Zope no Linux é simples. Você vai precisar da atual versão 2.3.4 do *Python* com suporte a *Zlib*. Embora o Zope seja escrito em Python em sua maior parte, alguns módulos foram implementados em C por razões de velocidade. Será preciso compilar o aplicativo antes de instalar. O arquivo tar.gz tem um script `configure` que gera automaticamente um `Makefile` para o processo de compilação e instalação.

As bibliotecas do Zope normalmente são instaladas em `/usr/local/Zope-3.0.x`. Obviamente, você pode mudar essa localização editando o parâmetro `--prefix` no script `configure`. Para iniciar uma instância do servidor, você precisa primeiro criar uma árvore de diretório para ela. Ela é usada não somente para armazenar o banco de dados do objeto para a instância do Zope (ZODB) como também para armazenar bibliotecas ancias exigidas especificamente para a instância. Claro que uma instalação do Zope pode usar diversas instâncias paralelas.

O script `mkzopeinstance` no diretório `bin` cria instâncias do programa. Se você especificar o caminho para a instância (cada instância fica em um diretório separado) e as credenciais para uma conta administrativa temporária, pode ativar uma instância digitando `runzope` (no diretório `bin` da instância). A configuração do servidor, as portas para HTTP e FTP e as várias opções de registro de eventos (*log*) estão disponíveis em `etc/zope.conf`. O arquivo de configuração usa o mesmo formato que o do servidor Apache. Por padrão, a instância do servidor web usará a porta 8080; o servidor de FTP usará a porta 8021.

Depois de iniciar o Zope digitando o comando `runzope`, pressione ***[Ctrl]+[C]*** para sair. Como isso é impraticável em uma aplicação rodando num servidor, é possível usar o *script* `zopectl` (também no diretório `bin` de cada instância) para controlar o servidor; o procedimento é similar ao do `apachectl` para o servidor web Apache.

Sobre o autor

Philipp von Weitershausen estuda física em Dresden, na Alemanha. Também é um desenvolvedor freelance de software e consultor e membro do grupo que desenvolve o Zope 3. Seu livro, ***Web Component Development with Zope 3*** *(Desenvolvimento de componentes para a web com o Zope 3) estará nas prateleiras européias em meados de 2005.*

Informações

[1] Plone: plone.org

[2] Zope: zope.org

[3] Silva: infrae.com/products/silva

[4] CPS: www.nuxeo.org/cps

[5] Zope X3: zope.org/Products/ZopeX3

[6] Five: codespeak.net/z3/five