

## Capítulo 11: Manipulando e Categorizando Tipos de Conteúdo

Tradutor: Michael Almeida Lucas de Souza (michael.almeida@sebrae.com.br)

Ao longo do livro, mostrei como adicionar conteúdo ao seu site e discuti os tipos de conteúdo que vêm com o Plone, como documentos, imagens entre outros. Até agora, entretanto, você esteve restrito à apenas estes tipos de conteúdo e os outros fornecidos por produtos que você pode achar na Internet. Mas, a parte mais poderosa do Plone é o assunto principal deste capítulo: manipulação destes tipos de conteúdo.

Neste capítulo compararei e contrastarei os diferentes tipos de conteúdo no Plone. Isto proporcionará algum “insight” nas táticas de desenvolvimento para os seus próprios projetos. Então, cobrirei os tipos de conteúdo e como são registrados internamente no Plone. A forma como são registrados é a base para customizar os tipos no formato que você quiser. Seguirei, então, para categorização e procura de conteúdo – tarefas que você vai querer saber como fazer. Preparado com este conhecimento, você será capaz de tomar importantes decisões de como desenvolver seu site e criar novos tipos de conteúdo.

Então, irei agora mostrar como manipular tipos de conteúdo. Uma vez que você puder customizar um tipo de conteúdo totalmente novo, você poderá ter usuários adicionando e editando praticamente tudo que você quiser! Alguns exemplos incluem:

- Usuários podem enviar uma imagem de uma cultura de células que são manipuladas usando bibliotecas de imagens e são então apresentadas ao usuário num certo formato de imagem.
- Usuários podem enviar um arquivo de áudio MP3, extrair o título e o artista do arquivo e colocar no Plone.
- Você pode construir uma loja e-commerce completa, deixando que os usuários do Plone adicionem itens como às roupas para venda, incluindo informações como frete, dimensões e garantia.
- Usuários podem enviar um documento do Microsoft Word e então manipulá-lo para remover certas partes do documento. Usuários com configuração de segurança reduzida podem ver apenas os documentos com as partes removidas.

Todas essas opções e outras estão disponíveis para você no Plone! Você realmente tem poucos limites. Esse é o motivo pelo qual Plone é um dos “frameworks” mais extensíveis e flexíveis disponível. O único limite real é a sua habilidade em programar em Python (ou poder pagar alguém que faça isso para você).

Neste capítulo, eu cobrirei tipos de conteúdo em detalhe incluindo como registrar e manipulá-los via Web. Mesmo que as partes seguintes não requeiram especificamente conhecimento sobre Python, eu recomendo que você pelo menos se familiarize com ele. Também incluo neste capítulo informações sobre formulários e como validá-los.

Este capítulo é requerido por qualquer um que queira desenvolver seus próprios tipos de conteúdo, não importando que se faça pela Web ou em Python, pois o capítulo cobre as áreas necessárias para entender registro de tipo de conteúdo.

O próximo capítulo continuará esta jornada, eu levarei você aos detalhes de como usar Python para escrever tipos de conteúdo. Em seguida você usará Archetypes para fazer à mesma coisa com um décimo de esforço e então fazer algumas coisas bem legais e avançadas com Archetypes. Mas agora, você irá direto aos tipos de conteúdo, partindo de cima!

## Uma Visão Geral sobre Tipos de Conteúdo

Para começar, eu estive usando certos termos ao longo deste livro. As explicações que eu dei têm sido um pouco superficiais, e é hora de expandir isto para melhorar o seu entendimento sobre eles. Os conceitos importantes são:

**Content type:** Este é um tipo de conteúdo registrado em um site Plone. Normalmente, mas não sempre, um tipo de conteúdo é algo que, dada certa permissão ao usuário, pode ser adicionado e editado através da interface do Plone. No Plone, é recomendável separar tipos de conteúdo como documentos, arquivos e imagens. Essa separação de conteúdo em diferentes tipos é um conceito básico do Plone.

**Item and object:** Estes termos referem-se à instância atual de alguma coisa, e são termos sobrecarregados, então suas definições geralmente dependem do seu contexto. Até agora eu usei estes termos para referir-me a alguma particular instância de um tipo de conteúdo, assim como um particular documento ou imagem. A partir de agora, eu usarei o termo específico, *content type*, para me referir ao atual tipo de conteúdo.

**Tool:** Uma ferramenta é um serviço que fica dentro de um site em Plone. Só irá existir uma instância de cada ferramenta num site em Plone. Na essência, a ferramenta não faz nada e os usuários externos da aplicação nunca saberão quantas ferramentas existem ou o que elas fazem. Porém, tipos de conteúdo ou requisições que os usuários podem fazer irão interagir com as ferramentas. Você já viu algumas ferramentas importantes *portal\_workflow*, *portal\_skins* e *portal\_actions*, por exemplo.

**Zope object:** Este é um objeto que fica dentro do Zope. Pode ser acessado via a Interface de Gerenciamento Zope (Zope Management Interface – ZMI) e provê funcionalidades ao usuário. Porém, não é algo que pode ser acessado ou controlado pelo Plone. Se você for a um site Plone e acessar o ZMI, você verá um grande número desses objetos no ZMI. Ferramentas são de um tipo de objeto, o site Plone é outro e os gerenciadores do cache são outros. Há uma grande similaridade entre estes objetos; por exemplo, Plone tem um tipo de conteúdo imagem, e o Zope tem um objeto imagem. Os dois executam tarefas similares e trabalham de forma semelhante, mas apenas um pode ser acessado via Plone.

**NOTE:** Apesar de que tudo dentro de um site Plone no Banco de Dados de Objetos Zope (Zope Object Database – ZODB) é um *Zope object*, eu uso este termo para descrever objetos que não são ferramentas ou instâncias de um tipo de conteúdo.

## Quando fizer Tipo de Conteúdo

Então, você está construindo sua grande aplicação em Plone que lhe trará fama e fortuna. Como estruturar isto, e como construir? Bom, isto depende do que você está construindo e do que você desenvolveu até agora. As questões seguintes podem ajudar você a decidir:

**\*\*Você está apenas mudando skins e comportamentos simples como os portlets?\*** Você pode fazer quase tudo que quiser, exceto escrever uma ferramenta ou um tipo de conteúdo em um skin. Você pode mudar todas as CSS, templates e scripts que vêm com o site Plone se você quiser.

**\*\*Membros do seu site irão adicionar múltiplas cópias deste item?\*** Se sim, então provavelmente você quer escrever um tipo de conteúdo.

**\*\* Este é um serviço que outros tipos de conteúdo podem usar?\*** Se sim, então provavelmente você quer escrever uma ferramenta.

**\*\* Você quer múltiplas cópias de algo, mas não quer que membros do seu site adicionem e editem isto?\*** Se sim, então provavelmente você quer um objeto Zope. Entretanto, você deverá repensar exatamente no que está fazendo.

O que normalmente acontece é que uma aplicação é desmembrada em alguns pedaços: uma ou duas ferramentas e um ou dois tipos de conteúdo. O capítulo 12 cobre a escrita de um tipo de conteúdo que tem como entrada código fonte, como um pequeno trecho de Python, e então faz destacamento de sintaxe. Se precisar disto em outros lugares, então você pode transformar isto em uma ferramenta que múltiplos tipos de conteúdo podem usar. Em resumo, ferramentas é a melhor forma para adicionar funcionalidade a um site em vez do que para algum tipo de conteúdo particular.

A definição para criação de um tipo de conteúdo é normalmente ditada pela necessidade que os usuários tenham de adicionar, editar, e controlar estes objetos. Pode ser tentador começar criar tipo de conteúdo para todo tipo de objeto, mas assim como qualquer desenvolvimento, você precisar ser cauteloso. Seria possível usar um tipo de conteúdo em vez de dois, com apenas algumas diferenças? Saber como configurar isto virá com a experiência, mas os próximos capítulos com certeza ajudarão.

## Configuração de Tipo de Conteúdo

Continuando, seu site Plone contém tipos de conteúdo, mas como o site Plone saberá que eles estão configurados? A resposta é que para cada tipo de conteúdo, seus atributos, métodos, segurança e skins são definidos em Python no sistema de arquivos e código associado. Esta informação é suficiente para o Plone entender como usar o produto. A única exceção a isto, como você já viu, é o “workflow”, que é normalmente definido externamente ao tipo de conteúdo. Alguns produtos têm seu próprio “workflow” que é adicionado ao tipo de conteúdo para o seu comportamento.

O capítulo 10 mostrou como os tipos de conteúdo são instalados no Plone através de um processo de dois passos: Primeiro, o produto é instalado no Zope. Segundo, o tipo de conteúdo é instalado em *cada* instância do Plone. O segundo

passo instala informações sobre o tipo de conteúdo, as quais vêm do sistema de arquivos e são, então, instalados no seu site Plone.

Por que de um processo de dois passos? No segundo estágio, é feita uma cópia localmente do produto do seu site Plone, e a partir daí você poderá controlar como o tipo de conteúdo se comportará para você. Quer um objeto de documento que tenha abas diferentes no topo? Quer um objeto de documento que seja manipulado diferentemente, tenha uma aparência diferente e até tenha um nome completamente diferente? Sem problema, você poderá modificar sua instância do Plone via Web.

Esta abordagem é a mesma que a do *portal\_skins*, onde você pode customizar uma skin na sua instância local. Quando ocorrem modificações no produto e você instala uma nova versão do Plone, essas mudanças afetarão o sistema de arquivos. Mas agora, você pode baixar e instalar essas mudanças; como você customizou-as no seu banco de dados, você manterá a versão customizada.

Cada tipo de conteúdo no Plone terá uma configuração na ferramenta *portal\_types*. Embora cada tipo de conteúdo na ferramenta *portal\_types* tenha apenas uma configuração, esse tipo tem um número ilimitado de objetos no seu banco de dados. A configuração é consultada quando necessário, então se você modificar a configuração, você atualizará todos os objetos desse tipo no banco de dados.

### Registro de Tipo de Conteúdo na Ferramenta *portal\_types*

Para acessar a informação sobre o registro, vá à ferramenta *portal\_types* no ZMI. Lá você encontrará uma lista com todos os tipos de conteúdo registrados neste site Plone. A maioria destes tipos de conteúdo é identificada como algo que você pode adicionar através de interface Plone com algumas exceções, como o Plone Site, TempFolder, e assim por diante.

Cada um destes objetos é uma instância de informação de tipo de fábrica, que é o nome para uma configuração de tipo particular. Clique em qualquer um destes objetos para acessar a informação do tipo; por exemplo, quando você clica em um evento, você terá uma cópia local da informação sobre o tipo de conteúdo. Você pode alterar isto via Web para modificar sua configuração. A seguir, estão os valores desse formulário:

- **Title:** Este é um título para o tipo de conteúdo.
- **Description:** Esta é a descrição que aparece para este tipo de conteúdo. Isto é usado se você for para o conteúdo da pasta e clicar em *adicionar* sem selecionar um tipo de conteúdo para adicionar, uma lista com todos os tipos de conteúdo e as suas descrições irá aparecer.
- **Icon:** Isto é o ID do ícone que é usado para este tipo de conteúdo.
- **Product metatype:** Este é um metatype para este tipo de conteúdo. Serve para equiparar o tipo de conteúdo do Plone com um metatype do Zope.
- **Product name:** É o nome do produto onde este metatype é definido.
- **Product factory method:** Este é um método que é chamado pela fábrica de produto para criar uma parte de conteúdo.
- **Initial view name:** Isto não é usado em Plone.

- **Implicitly addable:** Isto indica se este conteúdo pode ser adicionado ao Plone. Se for selecionado, então será adicionado a menos que especificado explicitamente de outra maneira.
- **Filter content types:** Se este tipo de conteúdo é uma pasta, então permita ele filtrar os tipos de conteúdo que podem ser adicionados por usuários a este objeto.
- **Allowed content types:** Se este tipo de conteúdo puder conter outros itens e *Filter content types* é permitido, apenas os tipos de conteúdos especificados na lista estarão permitidos.
- **Allow discussion:** Isto ajusta o status padrão para discussão de todos os tipos de conteúdos. Se for permitido, então os usuários poderão discutir o conteúdo. Quaisquer usuários poderão fazer isto dependendo da permissão *Discuss content*.

Você olhará agora alguns dos aspectos desta informação de registro em mais detalhes, incluindo alguns exemplos.

### Como Você Muda um Ícone para um Tipo de Conteúdo?

Como um exemplo, se você não gosta do ícone que aparece para um tipo de conteúdo, então é uma questão bem simples de enviar uma nova imagem e então de certificar-se de que o valor para o ícone é ajustado na forma previamente descrita. Os ícones trabalham melhor se tiverem um fundo transparente e forem 16 pixels de largura e 16 pixels de altura.

Clique *portal\_skins*, clique *custom*, e adiciona uma nova imagem. Então na ferramenta *portal\_types*, ajuste o valor para o ícone para ser igual ao ID do objeto enviado. Para testar que o ícone foi alterado, vá para interface do Plone e olhe aonde o objeto pode aparecer; por exemplo, faça uma procura ou olhe em *content add form*.

### Ações

Quando você está olhando para uma configuração de um tipo de conteúdo no *portal\_types*, você verá a aba *Actions*. Estas são as ações que podem ser feitas pelo tipo de conteúdo. Você olhou rapidamente ações no Capítulo 4, que contém uma lista detalhada de que a aba *Actions* possui.

#### *action*

Como você viu, ações são armazenadas em objetos da ferramenta. Muitas ferramentas contêm ações, mas você não tem o melhor modo para procurar a posição de uma ação. Se você quer mudar uma ação em particular no seu site Plone, você tem que encontrar a ferramenta que o armazena.

Uma vez que você encontrou a ação, você pode então modifica-la do modo que quiser. Por exemplo, se você quer adicionar uma nova ação como uma aba verde para um documento, você tem que procurar a posição perfeita. Os passos seguintes o ajudam a encontrar uma ação:

- Se você estiver procurando uma ação em uma parte do índice tal como *view* ou *edit*, então está em um tipo de conteúdo particular na ferramenta *portal\_types*.

- Se você está procurando uma ação para o site, então está na ferramenta *portal\_action*.
- Se você não conseguiu encontrar até agora, procure em uma ferramenta relacionada; por exemplo, *joining* e *logging* estão no *portal\_membership*.
- Se você não puder encontrar a ação que você está procurando após ter tentado os passos, vá para *portal\_actions* para ver a lista de ferramentas e olhe por todos que fornecem ações.

Plone olha as ações para os tipos de conteúdos da seguinte maneira:

- Para um objeto, são examinadas todas as ações.
- Para cada ação, as propriedades *conditions*, *permissions*, e *visible* são verificadas; se passarem, então as ações serão retornadas.
- Cada ação será mostrada na interface do usuário, normalmente na forma de abas no topo do índice ou no topo do site.
- A URL para esta ação é a URL do objeto com a atual *Action* adicionado ao fim.

Por exemplo, em um documento em *http://localhost.com/Plone/Document123*, a URL edição pode ser *http://localhost.com/Plone/Document123/document\_edit\_form*. Você deve observar valores importantes de edição de segurança para as propriedades *conditions*, *permissions*, e *visible* relacione para mostrar a ação na lista de ações. Isto significa se os usuários realmente quisessem, eles poderiam alterar a URL para ir para *http://localhost.com/Plone/Document123/document\_edit\_form* até mesmo se as permissões das ações não permitirem isto. Por esta razão, você deve sempre ter permissões nas ações atuais que serão executadas. Se você fosse um usuário que pode ver um objeto, mas não pode editá-lo, você ainda poderia alterar a URL para pegar o documento de editar formulário. Nenhum dano real foi feito ainda porque uma vez que você submeteu, a segurança seria verificada novamente e seria negado a permissão a você.

Ações normalmente são usadas em abas no Plone, mas desde que podem ser chamadas programaticamente, poderiam ser usadas de qualquer modo. Para chamar um ação programaticamente, você chama o método *listFilteredActionsFor* da ferramenta *portal\_actions*. Dado um objeto, isto retornará para você um dicionário Python para todas as ações de um objeto:

```
actions = context.portal_actions.listFilteredActionsFor(object)
```

Isto dá-lhe o seguinte:

```
{'site_actions': [
    {'category': 'site_actions', 'name': 'Small Text',
     'url': "javascript:setActiveStyleSheet('Small Text', 1);",
     'visible': 1, 'id': 'small_text',
     'permissions': ('View',)
    },

```

... e assim por diante

As abas verdes no topo são as combinações de duas categorias: *object* e *object\_tabs*. As ações retornadas do método dicionário de Python cujas chaves sejam os agrupamentos da categoria para essa ação. Assim, para capturar apenas os objetos ações para uma categoria, por exemplo, todas as ações na categoria *object* você poderia apenas acessar a chave do dicionário. Por exemplo, `actions["object"]` será dado a você uma lista de todas estas ações:

```
{'category': 'object',
 'name': 'Contents',
 'url': ' http://localhost:8080/Plone/folder_contents',
 'visible': 1,
 'id': 'folderContents',
 'permissions': ('List folder contents',)},
```

... e assim por diante

Você vai perceber que desde que você forneça o objeto que você está examinando, ele te levará para a ferramenta *portal\_types* e encontrar todas as ações para seu *portal\_type* particular, bem como quaisquer outras ações está pode ser relevante.

Se você quer adicionar uma nova aba para o tipo de conteúdo, tudo que você precisa fazer é ir para *portal\_types*, clicar no tipo de conteúdo, e selecionar a aba Ações. Então adicionar sua ação. Se a ação fosse para aparecer como uma aba verde para um tipo de conteúdo, então você teria que assegurar feito a categoria *object\_tabs*.

## Outros Objetos na Ferramenta *portal\_types*

Looking at the *portal\_types* tool, you'll probably notice you can add other object types to the folder such as DTML Method, External Method, Script (Python), and Scriptable Type Information. The first three of these options are present to provide support for the last option in the list, Scriptable Type Information.

Scriptable Type Information lets you define a type but create your own constructor permissions and construction script through the Web, instead of having them defined for you. If the default permission for a content type isn't sufficient, this may be an option. Although it's a useful-sounding option, I've never seen a good use for Scriptable Type Information over the standard factory-based type information, so don't worry about it.

## Armazenando a Informação do Tipo de Conteúdo no Sistema de Arquivo

Você tem visto agora como esta informação é armazenada em Zope, mas é naturalmente de algum lugar do sistema de arquivo. Está informação é

normalmente armazenada no produto em um dicionário, normalmente chamada *factory-based type information*. Listando 11-1 mostra a informação sobre a Pasta, que é o produto que mostra as pastas no Plone. Isto é retirado do arquivo *PloneFolder.py* localizado no diretório *CMFPlone*.

Listando 11-1. Fábrica-Baseada no Tipo de Informação.

```
factory_type_information = {
    'id': 'Folder',      'meta_type': 'Plone Folder',
    'description': """\
Plone folders can define custom 'view' actions,\
or will behave like directory listings without one defined.""",
    'icon': 'folder_icon.gif',
    'product': 'CMFPlone',      'factory': 'addPloneFolder',
    'filter_content_types': 0,
    'immediate_view': 'folder_listing',      'actions':
        ( {
            'id': 'view',          'name': 'View',
            'action': 'string:${folder_url}/',          'permissions':
(CMFCorePermissions.View,),          'category': 'folder',          }
        )
}
```

O dicionário Python traça os mapas dos formulários que você viu na interface Plone; por exemplo, *'meta\_type': Plone Folder* é o produto *meta\_type* e aparecerá neste campo. As ações aparecem como uma lista de dicionários para cada ação. Eu a pouco mostrei a primeira ação aqui, *View*, mas agora por esta informação deve ser familiar a você.

### Criando um Novo Tipo de Conteúdo de um Tipo Já Existente

*Repurposing* está levando a informação para um tipo de conteúdo existente e de criar múltiplo, copias ligeiramente diferentes do mesmo tipo. Se você quiser fazer um tipo que fosse quase o mesmo como um news item, mas não completamente, então *repurposing* possa ser uma opção rápida e simples.

O grande inconveniente desta aproximação é que você não pode mudar muito além das ações, os skins, e ajustar alguns tipos de conteúdos. Assim antes que você prossiga com esse caminho, por favor, esteja atento que você é limitado a estes pontos; você não pode adicionar novos campos ou atributos, por exemplo. Eu vi muitos e-mails na lista dizendo, 'Eu fiz isto muito, mas agora eu quero mudar os atributos da minha press release. Considere assim isto um aviso: Você não pode! Se você quiser fazer mais, confirme escrevendo tipos de conteúdo nos dois próximos capítulos.

Você quis fazer um tipo de press release que está como um item de notícia, mas faz o seguinte:

- Tem o nome *Press Release* na lista drop-down.
- Tem um ícone diferente.
- Tem um workflow diferente de um item de notícia.
- Tem uma vista diferente.
- Mantém a mesma estrutura de dados com o item de notícia.
- Retém o tipo do news item.

Bem, neste caso, repurposing um tipo de conteúdo é ideal. Para este exemplo, leve a informação de tipo factory-based para um item de notícia, carregue-o na ferramenta *portal\_types*, e então chame um press release. Isto permitirá que você reutilize todo o código existente e informações enquanto dando-lhe opções novas. Na ZMI, acesse *portal\_types* e complete os passos seguintes:

*Factory-based Type Information Press Release, Use default type information* Este é agora um exemplo da configuração para um news item, mas é chamado *Press Release*. Que vantagem isto lhe dá? Bem, você tem agora um outro tipo de objeto que pode ser adicionado pela Web por um usuário. Isto dá aos usuários de seu site uma maneira realmente fácil de distinguir entre um artigo de notícia e um boletim de impressão (press release), sem perder tempo com keywords ou metadata. Mostrará também nas buscas e em todos os outros lugares restantes como uma press release. Você pode agora mudar a configuração para boletim de impressão (press release), e sairá da configuração para o artigo de notícia intacto.

Mudando o ícone isto já foi discutido neste capítulo, simplesmente envie uma imagem para seu diretório de costume e então altere a propriedade do ícone na página *portal\_types* pra um boletim de impressão (press release). Se você for para *portal\_workflow*, você pode ver que cada tipo de conteúdo tem seu próprio workflow. Porque este agora é um novo tipo de conteúdo, você pode mudar o workflow para somente boletim de impressão (press release). Talvez os boletins de impressão (press releases) requeiram um estágio extra de revisão ou, quando publicado, emitir e-mails para determinados usuários. Você pode agora fazer um novo workflow, como eu descrevi no Capítulo 8, e atribui isto para seu boletim de impressão (press release).

Adicionando uma new view significa customizar a página template *newsitem\_view* e renomeando para alguma coisa significativa como *pressrelease\_view*. Você pode alterar este arquivo para adicionar alguma informação sobre a companhia no fim da página. Por exemplo:

```
<h2>About ACME Widget Company</h2>
```

```
<p>Our company is the prime maker of widgets in the world. Founded
in 1980 we've been providing excellent widgets to all parts of the
globe. For more marketing information, please contact: Joe Bloggs,
marketing director.</p>
```

Depois que você salvou suas mudanças para sua nova página template, retorne as configurações para o boletim de impressão (press release) no *portal\_types* e ir para página Actions. Mude a ação para ver um boletim de impressão (press release) que aponta para *newsitem\_view* para apontar para *pressrelease\_view*. Agora sempre que você ver um boletim de impressão (press release), está página view irá mostrar, como mostra a Figura 11-1.



Figura 11-1. Um exemplo de Python script carregado em Plone.

Neste caso eu adicionei um objeto boletim de impressão (press release), e o footer sobre ACME Company esta no template, então usuários não precisam lembrar de digitar isto todo tempo.

### Criando um Objeto Scripting

Uma vez que um objeto é registrado na ferramenta *portal\_types*, você pode então criar um objeto em seu site Plone. Você pode também certificar a criação do objeto programaticamente. Isto é útil para fazer objetos baseados em outros

fatores determinados ou criando objetos em massa. Plone possui 2 objetos Scripts (Python) úteis para isto:

**generateUniqueId:** Isto cria uma nova ID única para o tipo de objeto, por exemplo, *Folder.2003-12-19.7359814582*. Ele é único somente para a pasta criada dentro; se você criou um monte de objetos rapidamente, então é possível que eles não possam ser únicos.

**invokeFactory:** Isto pega um ID e coloca um nome. Criará um objeto de um tipo dado e dar um ID especificado.

Você fará um exemplo de um script que cria uma pasta e uma página padrão dentro da pasta, e nessa página padrão você vai colocar algum conteúdo específico. Se isto soar familiar, este é o que acontece quando você entra em um site e a pasta home é criada pra você. Os tipos de nomes são idênticos ao do registro dentro da ferramenta *portal\_types*, então se você deseja criar uma pasta e dentro criar um documento, você precisa passar os parâmetros *Folder* e *Document* para o script *invokeFactory*.

Listando 11-2 mostra um script que pega um ID único e cria uma pasta baseada nesta ID. Então isto passará dentro da pasta e criar um novo documento.

Listing 11-2. Pegando um ID e Criando uma Pasta.

```
##title=Create

##parameters=

# criar com um id aleatório

newId = context.generateUniqueId('Folder')

# criar um objeto de tipo Pasta

context.invokeFactory(id=newId, type_name='Folder')

newFolder = getattr(context, newId)

# criar um novo tipo de Documento

newFolder.invokeFactory(id='index.html', type_name='Document')

# tendo uma nova página

newPage = getattr(newFolder, 'index.html')

newPage.edit('html', '<p>This is the default page.</p>')

# retorne algo atrás para chamar um script

return "Done"
```

Se você adicionar isto como um objeto Script (Python) e testar isto usando a aba Test, você terá feito uma Pasta para você. Uma coisa interessante para notar é que isto cria uma pasta e documento no contexto atual, onde quer que o objeto do contexto possa ser.

## Registro do Tipo de Conteúdo

Eu mostrei uma variedade de modos para acessar o Plone, incluindo File Transfer Protocol (FTP) e WebDAV. Quando o Plone recebe conteúdo de uma dessas fontes, tem que lidar com o conteúdo de uma maneira apropriada. Este processo é executado pelo registro do tipo de conteúdo, que é visível no ZMI como a ferramenta *content\_type\_registry*. Se você visitar a ferramenta *content\_type\_registry* no Zope, você irá provavelmente ficar surpreso por mais uma forma mal elaborada na ZMI, mas não deixe isto te desanimar!

Quando um conteúdo é adicionado no Plone via FTP ou WebDAV, as regras no registro são executadas do começo para o fim, até uma combinação ser formada. A combinação é baseada no critério desta regra, e quando encontrado, o tipo de conteúdo apropriado para aquela regra é criado. Os seguintes são os quatro tipos diferentes de critérios:

**major\_minor:** Isto pega as duas partes (qualquer lado da contra-barras) de Multipurpose Internet Mail Extensions (MIME) tipo de arquivo que chegou e o compara. Se você encontrar qualquer parte em branco, então vai comparar tudo. Por exemplo, um *major\_minor* de *image* (isto é um espaço vazio no final) compara *image/jpeg*, *image/gif*, *image/png*, e assim por diante.

**extension:** Isto compara a extensão do nome de arquivo; cada extensão é separada por espaço. Assim, por exemplo, *doc pdf* compara *invoices.doc* e *report.pdf*.

**mimetype\_regex:** Isto desempenha uma expressão regular encontrada no tipo MIME. Por exemplo, *\_,^j* encontrou *image/jpeg*, *image/jpg*, *application/java*, e ai por diante.

**name\_regex:** Isto desempenha uma expressão regular encontrada no filename. Por exemplo, *^Invoice* vai encontrar *Invoice-123.pdf* mas não *Not\_an\_Invoice-123.pdf*.

Para adicionar um tipo, no fim da página, entre com o nome e a regra e o tipo de drop-down e clique Add. Isto vai criar uma regra no fim da página e permitirá que você incorpore um teste padrão que combine o tipo de regra que você criou e selecione o tipo de conteúdo que você quer criar da lista de drop-down. Então você pode clicar Up e Down para mover seu item para cima e para baixo, respectivamente, para aumentar sua importância.

Por exemplo, eu recentemente comprei uma câmera digital. Por causa do instalador Plone Windows que possui CMFPhoto e PIL já pronto, ela encontrou uma forma de mandar minhas fotos para um álbum online com pouco esforço. Primeiro, eu ativei o servidor FTP, e ai eu fui para o registro de tipo de conteúdo e fiz uma nova regra, baseada na extensão que mapeia *image/jpeg* para o tipo de conteúdo de foto. Então movi as regras acima das regras já existentes para imagens. A partir daí tudo que tive que fazer foi arrastar e soltar as fotografias dentro do meu cliente FTP, elas eram automaticamente baixadas no Plone, colocadas, e mostradas.

## Procurando e Categorizando Conteúdo

Você viu como você pode procurar por conteúdo no Plone, mas eu irei agora entrar em detalhe e mostrar como sobrescrever categorizando e procurando ocorrências de conteúdo. A ferramenta principal que armazena toda esta informação é chamada de *portal\_catalog*, que é uma versão um pouco diferente e estendida da ferramenta subjacente a ZCatalog. Você encontrará uma excelente referência online para o ZCatalog em [http://zope.org/Documentation/Books/ZopeBook/2\\_6Edition/SearchingZCatalog.stx](http://zope.org/Documentation/Books/ZopeBook/2_6Edition/SearchingZCatalog.stx).

O catálogo fornece três elementos chaves para o site Plone: Isto cria índices de conteúdo, prende a metadata sobre o conteúdo no índice, e fornece a interface de procura para rapidamente examinar o conteúdo de seu site Plone. De todos os objetos presentes em seu site Zope, apenas o exemplo atual de seu tipo de conteúdo são catalogadas. Objetos Zope, ferramentas, e outros objetos não são colocados no catálogo. Para está razão, a ferramenta catálogo é presa próxima aos tipos de conteúdo e a seu uso. Você pode acessar o catálogo acessando a ferramenta *portal\_catalog* na ZMI.

## Indexando Conteúdo

A primeira parte do trabalho do catálogo é construir indicadores do conteúdo. Um índice fornece primeiramente um método para rapidamente e eficientemente procurar o conteúdo. Para está razão, o conteúdo do índice não é projetado para ser claro ou fazer sentido; é especialmente projetado para fazer procuras rapidamente e eficientemente. Quando você faz uma procura em site Plone, você procura os indicadores, e o catálogo irá retornar combinando o resultado com a pergunta.

Um índice pesquisa um objeto Plone para um valor particular, um método, ou um atributo, e então indica o que o objeto retorna para a pergunta. A forma na qual ele indicia o conteúdo depende do tipo de índice. Tabela 11-1 lista todos os índices que vem com o Plone.

Tabela 11-1. Tipos Disponíveis de Índices.

Name	Description
<i>DateIndex</i>	Isto é projetado para posicionar datas, e isto deixa você fazer buscas por datas e horas.
<i>DateIndexRange</i>	Este é uma implementação mais eficiente de <i>DateIndex</i> para casos onde você tem duas datas, como começa e termina datas e fazendo várias procuras naquelas datas.
<i>FieldIndex</i>	Isto trabalha com cada resultado automaticamente e permite você procurar em o que quer que o conteúdo possa conter.
<i>KeywordIndex</i>	Isto pega a seqüência de palavras-chave e dividi separando-as. Irá retornar um resultado se qualquer palavra-chave no índice combinando com a pergunta dada. Isto é o ideal para procurar assuntos ou palavras-chave nos objetos.

*PathIndex* Isto indica o caminho de um objeto, como */Members/jane/myDocument*, como uma lista de objetos. Isto permite você perguntar o catálogo por todo conteúdo de *Members* sem ter que pedir a pasta. Um indicador *Path* irá retornar qualquer coisa abaixo da pasta *Members*.

*TextIndex* Isto é um antigo índice de texto que pega o texto, o divide, e o lista em forma de índice. Veja *ZCTextIndex*.

*TopicIndex* Isso constrói resultados pré-definidos no momento de catalogar. É usado para pedidos bastante requeridos.

*ZCTextIndex* Este é um novo índice que fornece capacidade de procura de texto-completo eficiente em partes de texto. Suporta um grande número de características, que serão discutidas depois em detalhes.

Você pode ver que os índices são definidos em catálogos clicando em *portal\_catalog* e selecionando a aba *Indexes*. Isto vai te dar a lista de todos os índices definidos no seu site Plone. As colunas são os nomes dos índices, do tipo, do número de batidas, de quando o índice foi modificado na última vez. Os tipos de indicadores foram cobertos momentaneamente, mas a Tabela 11-2 descreve que todos os índices padrões estão em um site Plone.

Tabela 11-2. Índices padrões que são ajustados no Plone.

#### **Name Type Description**

*Creator FieldIndex* Este é o nome do usuário que criou o objeto.

*Date FieldIndex* Está é a data efetiva; se não presente, é a última data modificada.

*Description TextIndex* O campo de descrição.

*SearchableText ZCTextIndex* A descrição, título, e corpo do objeto com um pedaço de texto que se pode procurar.

*Subject KeywordIndex* As palavras-chave para um artigo.

*Title TextIndex* Título do artigo.

*Type FieldIndex* O tipo portal como definido na ferramenta *portal\_types*.

*allowedRolesAndUsers KeywordIndex* Quem pode ver este conteúdo; este é um modo eficiente para examinar isto assim você pode filtrar a procura dos resultados

*created FieldIndex* Quando o item foi criado.

*effective FieldIndex* Quando um item será eficaz.

*end FieldIndex* Somente para eventos, quando o evento irá terminar.

*expires FieldIndex* Quando o item irá expirar e não será por muito tempo visível.

*getId FieldIndex* O ID para um item.

*id FieldIndex* Mesmo que *getId*.

*in\_reply\_to FieldIndex* Para discussões, dá o item que este comentário está respondendo.

*meta\_type FieldIndex* O metatype subjacente do item.

*modified FieldIndex* Quando o item foi modificado na ultima vez.

*path PathIndex* O caminho par o item.

*portal\_type FieldIndex* Mesmo que *Type*.

*review\_state FieldIndex* O estado de um objeto em in workflow.

*start FieldIndex* para eventos somente, quando o evento será iniciado.

Se você sempre inseguro do conteúdo de um index, então você pode ver o conteúdo dos indicadores na ZMI. Clique *portal\_catalog* e selecione Catalog, e isto irá listar todos os objetos catalogados neste tempo. Clique em object, para aparecer uma janela pop up com o conteúdo do índice e o metadata.

Para adicionar, remover, ou alterar o índice, retorne para aba Index. Utilize o Add drop-down box para adicionar um novo índice ou remover um índice. Se você quer executar um re-indexador de um índice particular, então selecione o indexes na esquerda e clique no botão reindex. Se você adicionar um índice para o catálogo, você então necessita de clicar no botão para se assegurar de que haja algum conteúdo em seu índice.

**NOTE** Se você possui um site grande, o re-indexamento pode ocupar muito tempo e consumindo processador, assim você pode evitar fazer isto durante horário de pico.

## Metadata

Quando o catálogo retorna um resultado, ele não retorna para você um, objeto; no lugar, retorna o metadata armazenado no catálogo. Este metadata é uma série de campos e colunas para cada valor no objeto. Do mesmo modo, um conjunto de lista de colunas para um site Plone é criado, como descrito na Tabela 11-3.

Tabela 11-3. Metadata Padrão Ajustado no Plone

### Name Description

*CreationDate* A data de quando o objeto foi criado.

*Creator* O nome do usuário da pessoa que está cria o objeto.

*Date* Esta é a data em operação; se não presente, é a ultima data modificada.

*Description* O campo de descrição.

*EffectiveDate* A data em operação.

*ExpiresDate* A data de término.

*ModificationDate* A data de modificação.

*Subject* As palavras-chave em um objeto.

*Title* O título do objeto.

*Type* O objeto *portal\_type*.

*created* Mesmo que *CreationDate*.

*effective* Mesmo que *EffectiveDate*.

*end* Para eventos somente, quando o evento terminará.

*expires* Quando o objeto irá terminar.

*getIcon* O ícone do objeto.

*getId* O ID do objeto.

*getRemoteUrl* usado somente para links; este é o URL apontado pelo link.

*id* Mesmo que *getId*.

*Location* Somente para eventos, onde o evento ocorrerá.

*meta\_type* O objeto é *meta\_type*.

*modified* Quando o objeto foi modificado.

*portal\_type* O objeto é *portal\_type*.

*review\_state* O estado do objeto no workflow.

*start* Somente para eventos, quando o evento ocorrerá.

### **Como um Objeto é Posicionado**

Tipos de conteúdos são posicionados automaticamente porque herdam de uma classe chamada *PortalContent*, o qual herda da classe chamada *CMFCatalogAware*. A classe *CMFCatalogAware* controla todo o código para assegurar que quando você adicionar, editar, cortar, copiar, deletar, ou renomear um objeto, os catálogos (e também workflow) são mantidos atualizados. Essencialmente o objeto é passado para o catálogo, e a instrução apropriada para o catálogo é chamada (o índice remove do índice, e assim por diante).

Dai o Catalogo então passa por cada índice e para cada índice pesquisado o objeto através da procura de atributos ou métodos no objeto. Para a maioria dos índices, o atributo ou método procurado é o mesmo nome que o do índice. Para o

nome do índice *Title*, ele poderia procurar por um atributo ou um método de nome *Title* e divulgar o índice com o resultado. Então ele repete o processo com cada coluna metadata.

Duas exceções para este processo são os tipos *FieldIndex* e *TopicIndex*. Quando você adiciona um *FieldIndex*, você pode especificar que o índice examina um valor diferente que o nome do índice. Por exemplo, você poderia fazer um índice com o ID *getVersion*, que procura o valor da versão. Como você verá mais tarde, alguns índices têm vantagens sobre outros, assim pode ser útil ter dois índices diferentes apontando para o mesmo valor.

*TopicIndex* é um tipo diferente de índice que gera uma série de ajustes ao mesmo tempo em que o conteúdo é indiciado. Se você deseja fazer muitas procuras para todas as imagens, então você poderia adicionar uma procura por *o.portal\_tipo==image*. Para fazer isso, você precisa criar um *TopicIndex* e então clicar em índice na aba de índices; Você pode até adicionar expressões múltiplas para gerar um índice. Neste momento, índices *TopicIndex* não são usados no plone.

### Como Você Re-indexaria Todo o Conteúdo em Seu Site Plone?

Se você fez um número grande de mudanças de nível de código, coloque um produto novo, renomeado ou movido na sua raiz do objeto Plone, então você pode necessitar re-indexar todo o conteúdo do seu site. Na ZMI, clique *portal\_catalog*, clique Advanced, e clique Update Catalog. Isto irá rodar o processo de atualização do seu catálogo.

**CAUTION** Isto é até mesmo uma tarefa mais intensa que re-indexar apenas um índice, e ele pode levar muito tempo e usar muita memória e processamento se você tem um banco de dados grande.

### Bases de dados relacionais vs. Plone

O desenvolvimento de tipo de conteúdo no Plone é um pouco diferente de desenvolver usando banco de dados relacional. Um paradigma comum no desenvolvimento nestes dias é LAMP uma combinação de Linux, Apache, MySQL, e PHP ou Perl. Neste paradigma, o dado é armazenado na tabela no banco de dados, e uma linguagem scripting fornece a camada de aplicativo para colocar o conteúdo fora do banco de dados e coloca-lo nos templates. Você procura o conteúdo enviando perguntas para o banco de dados no SQL, utilizando declarações *SELECT*.

Plone faz isto diferente utilizando um objeto do banco de dados. Todo artigo de conteúdo pode conter quaisquer atributos de qualquer tipo, e o banco de dados do objeto subjacente tenha cuidado de persistir aqueles atributos no banco de dados. Para procurar, todos os objetos então são posicionados na ferramenta *portal\_catalog*. Você tem que especificamente dizer ao catálogo exatamente quais atributos você tem que posicionar. Em vez de fazer chamadas do SQL, você usaria o catálogo para examinar os índices.

Esta diferença pode ser confusa no estágio do desenvolvimento, especialmente desde que os relacionamentos entre objetos não são criados e não são mantidos porque estariam em um banco de dados relacional. Ao invés, há duas maneiras

comuns de manter uma referência: utilizando um catálogo para manter a relação pelas palavras-chave ou outros valores usando uma pasta para o grupo de conteúdo. Archetypes, que eu discutirei no Capítulo 13, permitindo você manter relacionamentos facilmente. Isso se faz pelo catálogo.

## Procurando o Catálogo

Naturalmente, A grande pergunta é como procurar o catálogo e usar os resultados. A primeira destas tarefas depende dos índices, assim eu cubro cada um dos índices e mostro como procurá-los. A segunda tarefa envolve manipulação de resultados, assim eu mostro como você faz isto.

Todos os seguintes exemplos são em Python porque este é o melhor modo para procurar um catálogo. Eu também mostro um exemplo rápido de como capturar isto em uma página template. Eu recomendo utilizar Python para manipular o catálogo porque é realmente o melhor lugar para fazer coisas, permitindo você melhor flexibilidade sem ter que preocupar-se com a sintaxe.

No geral, você consegue procurar chamando o método `searchResults` no objeto `portal_catalog` e passando com uma série de parâmetros de palavra-chave. Existe um par de palavras-chave reservadas, mas o resto é mapeado diretamente para os índices com o mesmo nome. Assim se você quiser procurar o índice `SearchableText`, você passaria para o método que busca um parâmetro de palavra-chave para `SearchableText`. As palavras-chave reservadas são as seguintes:

- **sort\_on**: Este é o índice para classificar os resultados, supondo que o índice permita classificar (os índices full-text não permitem classificar).
- **sort\_order**: Isso é *reverse* ou *descending*; se não especificado, o padrão é *ascending*.
- **sort\_limit**: Esta é uma sugestão de otimização para fazer a classificação de um pouco mais rápida.

Assim, uma procura geral para todos os itens que mencionam Plone e são publicados na ordem *Date* procurando alguma coisa assim:

```
context.portal_catalog.searchResults(
    review_state = "published",
    SearchableText = "Plone",
    sort_order="Date"
)
```

A busca irá retornar a intersecção do resultado do índice, assim isto vai encontrar todos os itens que mencionam Plone *and* são publicados. Você não pode fazer buscas que não são a união de resultados; entretanto, você poderia fazer resultados múltiplos e então adicionar os resultados juntos, mas este caso é muito incomum, de qualquer forma.

**TIP** Se você fizer uma procura sem valores, então os conteúdos inteiros do catálogo são retornados. Por padrão, todas as buscas adicionam valores para efetivo e datas de fim, assegurando que você veja o conteúdo somente entre estas vezes, a menos que o usuário que está fazendo a busca tenha a permissão *Access inactive portal content*.

### Procurando um Campo ou um índice da Data

Para procurar um *FieldIndex*, passe com o valor do campo. Qualquer resultado que combine será retornado; por exemplo, procurar por todas as imagens em um site, utilize o seguinte:

```
results = context.portal_catalog.searchResults(
    Type = "Image"
)
```

Um campo de índice pode tomar uma quantidade de objetos também, e o índice vai tentar encontrar todos os valores dentro deles através de uma comparação de valores. Esta quantidade poderia ser de duas datas, dois números, ou dois strings, isto depende realmente do valor do *fieldIndex*. Você faz isso passando um dicionário para o índice, em vez de apenas string. O dicionário deve conter dois valores: Uma lista chamada *query*, a qual contém os valores a serem testados, e o *range*, que determina a quantidade de valores. O *range* é uma string do seguinte:

- **min**: Qualquer coisa maior que o item menor.
- **max**: Qualquer coisa menor que o item maior.
- **minmax**: Qualquer coisa menor que o maior, e maior que o menor.

Por exemplo, para encontrar todos os eventos que tem um tempo de início maior que agora (em outras palavras, qualquer coisa no futuro), utilize o seguinte:

```
from DateTime import DateTime

now = DateTime()

results = context.portal_catalog.searchResults(
    Type = "Event"
    end = { "query": [now,],
           "range": "min" }
)
```

Para procura em uma escala, tal como todos os artigos de notícias em Dezembro, você precisará para calcular o início e o fim das datas para o mês. daquelas datas, você pode construir a seguinte pergunta:

```
from DateTime import DateTime
```

```

start = DateTime('2004/12/01')

end = DateTime('2004/12/31')

results = context.portal_catalog.searchResults(
    Type = "News Item",
    created = { "query": [start, end],
                "range": "minmax" }
)

```

Índices de datas trabalham da mesma maneira como índices de campos, e frequentemente você verá datas colocadas dentro de índices de campos, que trabalham bem.

### Procurando um `KeywordIndex`

Por padrão, um `KeywordIndex` retorna todos os valores que combinam no índice de palavra-chave. `Subject` é o único `KeywordIndex`; está é a palavra-chave que o usuário atribuiu para um objeto através da aba `Properties` da interface `Plone`. Para procurar por todos os artigos como a palavra-chave *Africa*, use isto:

```

results = context.portal_catalog.searchResults(
    Subject = "Africa"
)

```

Similar para o `FieldIndex`, o `KeywordIndex` pode ser passado a uma pergunta mais complicada, com diversos objetos e um operador `and/or` (*or* é padrão). Isto permitiria que você encontrasse todos os objetos que têm quase toda a combinação das palavras-chave. Para encontrar todos os objetos que tem o assunto *Africa* e *sun*, use o seguinte:

```

results = context.portal_catalog.searchResults(
    Subject = { "query": ["Africa", "sun"],
                "operator": "and" }
)

```

### Procurando um `PathIndex`

Um caminho de índice permite você procurar por todos os objetos em um determinado caminho. Retornará todo objeto abaixo na localização atual, assim se você pedir por todos os objetos em *Members*, retornará tudo e todos diretórios do home. Por exemplo, para todos objetos que tem *Members* em seu caminho, use isto:

```
results = context.portal_catalog.searchResults(
    path = "/Plone/Members"
)
```

Se você quiser avançar restrinja este, você pode fazer passando através de um nível de parâmetro que ajusta onde você espera o valor para ser. O nível é o número que representa sua posição no caminho, da esquerda ao dividir isto por contra-barras. Por exemplo, no código precedente, *Plone* é nível 0, *Members* é nível 1, e assim por diante. Similarmente para *KeywordIndex*, você pode passar através um operador and/or. Para pegar todos os objetos na pasta */Plone/Members/danae* e a pasta */Plone/testing/danae*, utilize o seguinte:

```
results = context.portal_catalog.searchResults(
    path = { "query": ["danae"],
            "level" : 2 }
)
```

### Procurando um Índice ZCText

*ZCTextIndex* é o mais complicado de todos índices e is the most complicated of all indexes and takes a whole host of options. Cada *ZCTextIndex* requiere um lexicon; felizmente, Plone cria e configura tudo isso fora da caixa. Se você clica *portal\_catalog*, selecione a aba Contents, e clique *plone\_lexicon*, você pode ver a configuração padrão do lexicon. Clicando na aba Query irá mostrar pra você todas palavras que estão no lexicon construído fora do conteúdo do seu site Plone.

O *ZCTextIndex* é procurado utilizando o formato que eu descrevi no Capítulo 3. faz condições semelhantes para a busca que você utiliza no Google ou em outros mecanismos de busca. No seu mais básico, você pode buscar por qualquer termo (note que isto é case insensitive), tipo assim:

```
results = context.portal_catalog.searchResults(
    SearchableText = "space"
)
```

Mas você pode também por tudo o que segue:

- **Globbing:** Use um asterisco para significar qualquer letra. Por exemplo, *tues encontra tuesday etuesdays\**. Você não pode usar asteriscos no começo da palavra, You can't use the asterisk at the beginning of a word, de qualquer forma.
- **Single wildcards:** Utilize uma interrogação para significar uma letra. Por exemplo, *ro?e encontra rope, rote, role*, e assim por diante. Você não pode utilizar interrogação no começo da palavra.

- **And:** Utilizando *and* significa que ambos os termos no outro lado dele devem existir. Por exemplo, *rome and tuesday* irão retornar somente um resultado com ambas as palavras que estão no conteúdo.
- **Or:** Utilizando *or* significa que um ou outro termo pode existir. Por exemplo, *rome or tuesday* irão retornar um resultado se qualquer uma daquelas palavras estiver no conteúdo.
- **Not:** Utilizando *not* retorna o resultado onde este não está presente (o prefixo *and* é requerido). Por exemplo, *welcome and not page* poderia retornar resultados para páginas que contém *welcome*, mas não *page*.
- **Phrases:** Você pode agrupar frase com aspas duplas (") e significar diversas palavras uma após a outra. Por exemplo: "*welcome page*" encontra *This welcome page is used to introduce you to the Plone Content Management System*, mas não *Welcome to the front page of*.
- **Not phrase:** Você pode especificar uma frase com o prefixo menos (-). Por exemplo, *welcome -"welcome page"* encontra todas páginas com *welcome* nelas, mas não uma que combina com a frase *welcome page*.

**TIP** Se você executar uma procura sem texto, então nenhum resultado é devolvido.

### Utilizando os Resultados

Assim você tem alguns resultados, agora o que você vai fazer com eles? A primeira coisa que muitas pessoas fazem é olhar aos resultados e assume que é uma lista dos objetos que foram catalogados. Bem, isto não é; muito, é uma série de catalog brains. Estes brains são objetos que contêm as colunas de metadata definidos mais cedo. Você pode acessar qualquer coluna como se fosse um atributo. Por exemplo, para imprimir todos os IDs dos objetos de resultado, use o seguinte:

```
results = context.portal_catalog.searchResults()

for result in results:

    print result.getId

return printed
```

Neste exemplo, *getId* estão nome de uma coluna metadata, assim vai aparecer o valor *getId* que o catálogo teve para aquele objeto. Se você tentar acessar um valor que não existe como uma coluna metadata, então você adquirirá um *AttributeError*. Os seguintes são poucos métodos disponíveis que são úteis:

- **getPath:** Isto retorna o trajeto físico para este objeto dentro de Zope.
- **getURL:** Isto retorna a URL para este objeto com hospedagem virtual aplicada.
- **getObject:** Isto retorna o objeto atual.
- **getRID:** Este é um ID único para o objeto no catálogo, e muda cada vez que o objeto não é catalogado. É para finalidades internas somente.

Assim, se você quiser adquirir o objeto para cada resultado, você pode fazer assim, como verá no exemplo seguinte. Entretanto, existe uma razão para qual o catálogo não faz isso, é muito pesado (em termos da computação) porque isso envolve em buscar objetos do bando de dados (e todos os objetos entre eles) e

fazer várias checagens de segurança. Se você puder tentar fazer seu metadata conter a informação correta, você terá uma aplicação muito mais rápida. Obviamente, às vezes o metadata não pode conter tudo, mas é importante considerar no projeto. Para pegar cada objeto, use utilize o seguinte:

```
results = context.portal_catalog.searchResults()

for result in results:

    object = result.getObject()

    print object

return printed
```

Considerando que você tem uma lista de Python destes brains, isto agora é direto para manipular os resultados em uma maneira que você vê ajuste. Para encontrar quantos resultados foram retornados, você pode apenas chamar a função *len* na lista, como assim:

```
results = context.portal_catalog.searchResults()

print "Number of results", len(results)

return printed
```

**NOTE:** *len* é uma função Python que fala pra você o tamanho de um item.

Para pegar apenas os primeiro dez itens, utilize parte do Python, como assim:

```
results = context.portal_catalog.searchResults()

return results[:10]
```

Para fazer mais filtrado, você poderia filtrar manualmente a lista inteira, tipo assim:

```
results = context.portal_catalog.searchResults()

for result in results[:10]:

    # Title returns a string so we can use the find method of

    # a string to look for occurrence of a word

    if result.Title.find("Plone") > -1:

        print result.Title

return printed
```

Para pegar um objeto aleatório de um catálogo, utilize um módulo *random*, tipo assim:

```
import random

results = context.portal_catalog.searchResults()

r = random.choice(results)

object = r.getObject()

return object
```

### Juntando Tudo: Fazendo um formulário de Pesquisa

Na discussão precedente, eu mostrei como pegar resultados fora de um catálogo, e utilizei objetos Script (Python) para demonstrar isto. Mas você provavelmente está se perguntando, como eu posso fazer isto de uma página template?

Eu começarei de outro lado e primeiro assumirei que você tenha os resultados de uma pergunta do catálogo e passarei por eles na página template utilizando *tal:repeat*. Isto é como muitos portlets são colocados juntos e publicados, eventos portlets ambos apenas fazem perguntas e então mostram os resultados. Aqueles portlets encaixam a pergunta na página template chamando ele diretamente:

```
<div tal:define="results python:
here.portal_catalog.searchResults(Type="Event")">
```

Ou chamando um objeto separado do Script objeto (Python) que retorna o resultado. Por exemplo, no seguinte, o script é chamado *getCatalogResults*:

```
##parameters=

kw = {}

# enter your query into the kw dictionary

return context.portal_catalog(**kw)
```

Na página template, você pega os resultados da seguinte maneira:

```
<div tal:define="results here/getCatalogResults">
```

Após ter feito isto, você precisa dar um loop nos resultados utilizando a sintaxe padrão *tal:repeat*. Você pode acessar cada coluna metadata diretamente na Template Attribute Language (TAL) fazendo uma expressão do caminho para a coluna. Assim, dando um brain, você pode pegar o título da metadata chamando *result/Title*. Lista 11-3 mostra uma página de exemplo de loops que passam o conteúdo de *getCatalogResults* e mostra cada item em uma simples lista desordenada.

Lista 11-3. Looping através de *getCatalogResults*

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en-US"
```

```

    lang="en-US"

    metal:use-macro="here/main_template/macros/master"

    i18n:domain="plone">

<body>

<div metal:fill-slot="main">

<ul tal:define="results here/getCatalogResults">

    <li tal:repeat="result results">

        <a href=""

            tal:attributes="href result/getURL"

            tal:content="result/Title" />

            <span tal:replace="result/Description" />

        </li>

</ul>

</div>

</body>

</html>

```

Uma propriedade do método *searchResults* é que você não passa algum parâmetro para a função, ele irá procurar através do pedido enviado. Então se você quer permitir uma forma de colocar parâmetros para seus resultados, então tudo o que tem a fazer é mudar a linha de resultados anteriores para o seguinte:

```

<ul tal:define="
    results python: here.portal_catalog.searchResults(REQUEST=request)
">

```

Você pode refazer sua pergunta e pode juntar qualquer índice à URL. Por exemplo, se você chamar esta página template *testResults* e anexar *?Type=Document* para o final da URL do seu navegador, somente os documentos do seu site podem aparecer. Já que você pode passar quase qualquer valor pedido, você pode ajustar um formulário de procura que poderia passar essa informação pelo formulário de busca. Este é o que faz procura e procura avançada de páginas; você vai notar que ir para um site Plone e procurar por *beer* em caixa de procura, sua URL agora irá possuir *?SearchableText=beer*.

Assim, Lista 11-4 mostra uma forma para chamar sua página template.

## Lista 11-4. A Forma Para Chamar Seu Template

```

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en-US"
      lang="en-US"
      metal:use-macro="here/main_template/macros/master"
      i18n:domain="plone">
<body>
<div metal:fill-slot="main">
  <p>Select a content type to search for</p>
  <form method="post" action="testResults">
    <select name="Type">
      <option
        tal:repeat="value python:here.portal_catalog.uniqueValuesFor('Type')"
        tal:content="value" />
    </select>
    <br />
    <input type="submit" class="context">
  </form>
</div>
</body>
</html>

```

Este script utiliza o método chamado *uniqueValuesFor* no catálogo, qual retornará todos os valores únicos que existem para um índice. Isto o deixa executar uma tarefa tal como uma pequena drop-down box em um formulário, que é algo útil de se possuir.

Neste momento, se torna um exercício em HTML e página templates para fazer as páginas é tão complicado como você gostaria. Claro que, o melhor lugar para procurar tudo isto está nos templates atuais do Plone, que dão linhas sob linhas de grandes exemplos. Todos os portlets com os quais você está familiarizado no plone (como os de calendário, eventos, relacionado, assim por diante) são todos construídos usando requerimentos de catalogo para determinar o que mostrar.

Neste capítulo, eu forneci para você com uma vista geral das maneiras de desenvolver um site Plone e como trabalhar com tipos de conteúdos no seu site. Eu demonstrei com um tipo de conteúdo é construído e então referenciado através do catálogo. Está é uma metodologia chave para desenvolvimento em Plone.

No próximo capítulo, mostrarei como desenvolver um novo tipo de conteúdo eu vou te mostrar como desenvolver um tipo de conteúdo bem do começo. Você vai ver como voce pode integrar um novo tipo de conteúdo com o registro de catalogo na ferramenta *portal\_types*