Construindo portais com Plone

Módulo 9

Dexterity

Dexterity TTW

Módulo 9

Dexterity

Dexterity >> O que é?

Framework para incorporação de tipos de dados ao Plone.

Sucessor do Archetypes.

Mais rápido, modular e menos complexo.

Foi criado para servir duas audiências: Administradores e desenvolvedores.

Dexterity >> Para administradores

Habilidade de se criar tipos de conteúdo pela Web (TTW).

Habilidade de se ligar e desligar vários aspectos (chamado “behaviors”) dos tipos de conteúdo.

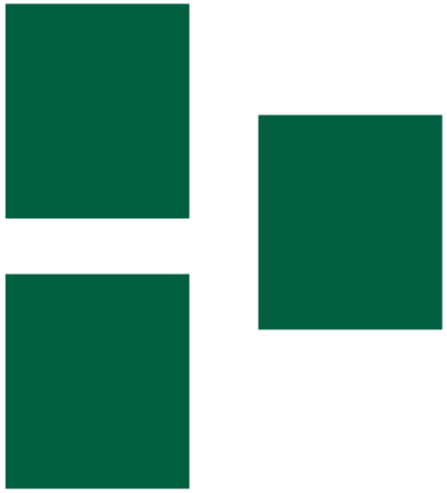
Dexterity >> Para desenvolvedores

Habilidade para se criar tipos de conteúdo mais fácil, rápido e com menos código gerado automaticamente.

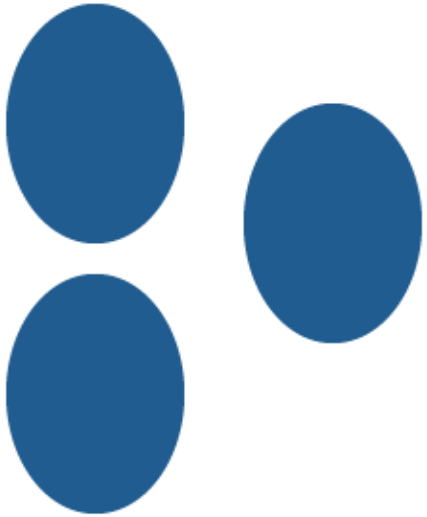
Tipos de conteúdo com performance aprimorada.

Tipos de Conteúdo

Representam instâncias de conteúdo com características em comum.



Páginas



Alguma outra coisa

Tipos de Conteúdo (2)

Um tipo de conteúdo no Plone pode ser ou um *container* ou um *item.* (As vezes chamados de *folderish* e *non-folderish)*.

Uma relação do tipo *one-to-many* geralmente é modelada com um *container* (o *one*) contendo vários itens (os *many*). Entretanto, também é possível usar referências. Exemplo: Projeto (Container) e Tarefas (Itens)

Todo tipo de conteúdo possui um schema.

Tipos de Conteúdo > Schema

É um conjunto de campos com propriedades relacionadas a eles como *title, default value, constraints*, etc.

O schema é usado para gerar formulários e descrever instâncias de um tipo.

Dexterity >> Componentes

Schema

Conjunto de campos e atributos do seu tipo de dados.

Ex: Campos texto, data, rich text, etc.

Behavior

Conjunto de comportamentos do seu tipo de dados.

Ex: metadados, categorização, range de datas etc.

FTI – Factory Type Information (veremos a seguir)

Dexterity > Through The Web

Ao instalar o pacote Dexterity pelo buildout ele já fica disponível em Site Setup > Dexterity Content Types.

Dispõe de todas as facilidades do desenvolvimento TTW.

Para migrarmos tipos criados TTW para o sistema de arquivos precisamos exportá-lo.

Mostrar

Pacote Dexterity

Módulo 9

Dexterity

Exercício 3 > Instalando o pacote

Vamos utilizar um esqueleto chamado dexterity para gerar o nosso pacote, em src:

../bin/zopeskel dexterity treinamento.content

(Aperte [Enter] para todas as perguntas)

Adicione o novo pacote ao buildout.

Rode o buildout (-c develop.cfg)

Reinicie a instância e verifique se o novo pacote já está disponível para ser instalado.

Exercício 4 > Mudando o título do nosso pacote

Na raiz do seu pacote (treinamento.content/treinamento/content), edite o arquivo configure.zml:

title="Example Dexterity Product”

title=“Projeto Alfa – Tipos”

description="Extension profile for Example Dexterity Product”

Description=“Tipos de conteúdo Dexterity para o Projeto Alfa”

Reinicie a instância e verifique as modificações.

Dexterity > Design de Tipos

Quando se está resolvendo um determinado problema com o Plone, frequentemente se implementa novos tipos de conteúdo.

Neste módulo, vamos implementar um conjunto de tipos para auxiliar organizadores de conferências.

Queremos gerenciar a programação da conferência, que consiste de várias palestras.

Cada palestra terá um título, descrição, duração e um palestrante.

Também queremos gerenciar a biografia de cada palestrante.

Dexterity > Design de Tipos (2)

Um tipo de conteúdo **Palestrante** será usado para representar a biografia do palestrante. Campos incluem nome, descrição e experiência profissional.

Um tipo de conteúdo **Programa** representará a programação da conferência. Além dos metadados básicos, ele listará os períodos disponíveis para adição de palestras. Esse tipo será folderish.

Um tipo de conteúdo **Palestra** representará a palestra. Palestras só podem ser adicionados dentro de **Programas**. Um conteúdo do tipo Palestra conterá informação sobre a palestra e permitirá ao usuário escolher a hora e o palestrante.

Criando um tipo conteúdo

Dentro de $INSTANCE/src/treinamento.content

./../../bin/paster addcontent dexterity_content

Criaremos dois tipos:

- Palestra
  - Name: Palestra
  - Description: Uma palestra em uma conferencia
  - Global allow: False
- Programa
  - Name: Programa
  - Description: A programacao de uma conferencia
  - Folderish: True
  - Global allow: True

Criando um tipo conteúdo (2)

Observe que decidimos colocar o tipo Programa como container, pois vamos colocar Palestras dentro dele.

O global-allow de Palestra colocamos False, pois queremos que Palestras sejam adicionadas apenas dentro de Programas.

Ainda falta criarmos o tipo Palestrante, faremos isso mais à frente.

Mostrar os principais arquivos gerados pelo **paster**.

Exercício 6: O Schema de Programa

Exercício 6

Exercício 7: O Schema de Palestra

Exercício 7

Atenção, para fixar melhor a definição de schemas, tente escrever o schema sem consultar a solução do exercício.

Schemas Interfaces

Um schema é, basicamente, uma interface (zope.interface.Interface) com campos.

Os campos padrão podem ser encontrados no pacote zope.schema (veja os imports).

Configurações de Tipo

Para cada tipo, existe um XML em profiles/default/types/

Esse XML é usado para realizar configurações no tipo.

As linhas mais importantes (abra palestra.xml):

- title e description
- global_allow
- add_permission
- behaviors
  - Aspectos reusáveis que provêem semântica e/ou campos.
  - INameFromTitle: id derivado do título do objeto (comportamento padrão)
  - IBasic: Adicionaria os campos title e descrition (não queremos isso, já que criamos os nossos) (Exercício 8: Remover)

Configurações de Tipo (2)

Abra programa.xml dentro de profile/default/types

Vamos permitir a adição apenas de **Palestras** dentro de **Programas**.

Exercício 9:

```
<property name="filter_content_types">True</property>
<property name="allowed_content_types">
   <element value=”treinamento.content.palestra" />
</property>
```

Criando o tipo conteúdo Palestrante

Dentro de $INSTANCE/src/treinamento.content

./../../bin/paster addcontent dexterity_content

Palestrante

- Name: Palestrante
- Description: Uma pessoa apresentando uma palestra na conferencia
- Folderish: False
- Global allow: True
- Allow_discussion: False

Campos do Tipo Palestrante

Não usaremos Schema para o tipo Palestrante. Ao invés disso, usaremos modelos XML. Com isso, poderemos editar o tipo TTW, e exportar os campos para usar no sistema de arquivos.

Exercício 11: Tipo Palestrante TTW

Desative o comportamento “Metadados Dublin Core”.

Campos:

- Nome -> (title) -> TextLine
- Sumário -> (description) -> Text
- Biografia -> (biografia) -> required=False -> RichText
- Foto -> (foto) -> required=False -> Image

Exercício 12: Migrando para o sistema de arquivos

Exercício 12

Perguntas?