Construindo portais com Plone

Módulo 3

Templates

Templates

- Quando o Plone vai mostrar uma página, duas perguntas precisam ser respondidas:
  - O que será mostrado? Isto é, quais conteúdos serão buscados do banco de dados?
  - Como será mostrado?

- Os templates permitem responder a essas duas perguntas.
- São modelos de páginas. Quando aplicados a um ponto do site (um objeto), mostram o que se quer mostrar.

Zope Page Templates (ZPT)

- É a linguagem de templates do Zope.
- Gera páginas web dinâmicas (através de HTML).
- Facilita colaboração entre programadores e designers (separação)
- É HTML válido e interage naturalmente com qualquer editor HTML.

Zope Page Templates (ZPT)

- PHP, ASP, JSP são linguagens que geram páginas dinâmicas, por que inventar outra?
- Nenhuma dessas linguagens leva em consideração o designer web.
- Facilita especialização das atribuições (designers em templates e programadores em python).
- O fato de gerar HTML válido proporciona a possibilidade da utilização de editores WYSIWYG e desenvolvimento TTW.

Templates >> Exemplo

- Criar um objeto do tipo Link chamado Yahoo.
- Acessar o template base_view no browser:
  - http://..../yahoo/base_view
  - Isto é, template view está sendo aplicado em obj.
- Fazer o mesmo com base_edit.
- Acessar http://..../yahoo/base_view/manage.
  - Visualizar o código do template.

Zope Page Templates (ZPT)

- É a linguagem de templates do Zope.
- Permite inserir comandos e diretivas no código HTML das páginas para torná-las dinâmicas.
- Se divide em duas partes: TAL e METAL.

Template Attribute Language (TAL)

- Atributos especiais adicionados às tags HTML.

- **content**: Substitui o conteúdo de um elemento.
  - Titulo

    Saída HTML:

    ** Este Site **

Template Attribute Language (TAL)

- **replace**: Substitui o elemento por completo.
  - Titulo

Saída HTML:

Este site

Criação de templates pela ZMI

- Para criar um template é preciso fazer o seguinte:
  - Abrir a ZMI.
  - Navegar até portal_skins → custom.
  - Selecionar “Page Template” e clicar em Add.
  - Entrar com o ID do template e clicar em “Add and Edit”.
  - Usar a aba Test para testar.

Templates >> Exercício 1

- Consulte o enunciado do exercício no material.

Templates >> Exercício 2.1

- Consulte o enunciado do exercício no material.
- Faça apenas o exercício 2.1

Template Attribute Language (TAL)

- **condition**: Permite que uma condição seja testada a fim de poder exibir ou não o um elemento HTML.

  - Teste

  Saída HTML:

  Se expressao = True:
  ** Teste **
  Se expressao = False:
  (Nada)

Template Attribute Language (TAL)

- **define**: Define variáveis.
  -
    Título

- **repeat:** Itera em uma lista (loop).
  - <ol tal:repeat="obj objetos">
    <li tal:content="obj/Title">Título

Templates >> Exercícios 2.2 e 2.3

- Consulte o enunciado do exercício no material.
- Faça os exercícios 2.2 e 2.3

Template Attribute Language (TAL)

- **attributes**: Substitui atributos de um elemento.
  - <a href="#"
    tal:attributes="href context/absolute_url">
    Clique aqui

  Saída HTML:

  Clique aqui

TAL Expressions (TALES)

- Expressões usadas nos comandos TAL.
- Exemplo, no comando define:

  <p tal:define="nome **string:Luiz**”> </p>

TAL Expressions (TALES)

- Os principais tipos de TALES são:
  - Path: um caminho até um objeto do site, ou o nome de uma variável:
    - ✓ minha-pasta/meu-objeto
    - ✓ Caracteriza-se pelo uso de “/”.
    - ✓

TAL Expressions (TALES)

- String: um literal String:
  - string: palavra
  - <p tal:define="nome **string:Luiz**”>
- Not: a negação lógica de uma outra expressão:
  - not:expressao
  -
- Python: um trecho de código Python:
  - python: 1 + 2
  -

TAL Expressions (TALES)

- Outras expressões recorrentes:
  - request/URL – Requisição URL corrente.
  - user/getUserName – Nome do usuário.
  - container/objectsIds – Ids do container atual.

TAL Expressions (TALES)

- Expressões podem ser combinadas com o operador “|”. Caso a primeira expressão seja falsa ou vazia, a segunda é utilizada:
  - title | id
- Caso comum do uso de “|” é o nothing.
  - <span tal:content=“nome | nothing”>

ZPT Variáveis Internas

- Além do nothing temos algumas outras variáveis internas no ZPT:
  - context – contexto, o objeto ao qual se aplica o template.
  - request – representa a requisisição feita ao servidor.
  - user – representa o usuário logado atualmente.
  - E outros menos importantes…

Declarações múltiplas de TAL

- Pode-se usar várias declarações TAL em uma mesma tag HTML, porém é preciso atentar para:
  - Somente uma declaração de cada tipo.
  - tal:replace e tal:content conflitam.
  - A ordem das declarações não importa, a ordem de execução dos TALES será sempre:
    - define, condition, repeat, replace/content, attributes.

Declarações em partes de TAL

- Algumas tag TAL suportam declarações em bloco.
- tal:define e tal:attributes pode ser declarados em várias partes separadas por “;”.
- Exemplo:
  - <div tal:define=“url context/absolute_url;
    name user/getUserName;
    id user/getId;
    palavra string: palavra”

Structure

- tal:replace e tal:content convertem a informação que recebem para “texto plano”.

- É uma palavra reservada que apresenta o conteúdo de uma variável em HTML.

- Solução para quando se quer exibir, na página, o conteúdo todo da variável incluindo qualquer tag que possa haver.

- Exemplos adiante…

Templates >> Exercício 2.4

- Consulte o enunciado do exercício no material.

Templates >> Exercício 2.5

- Consulte o enunciado do exercício no material.

Templates >> Exercício 3

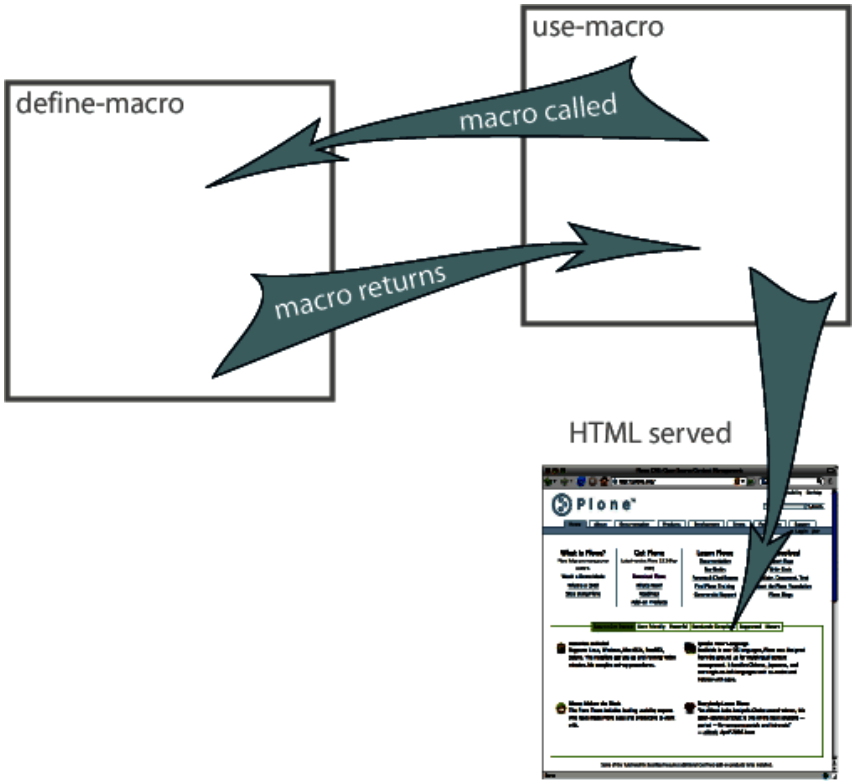
- Consulte o enunciado do exercício no material.
- Faça os exercícios 3.1, 3.2 e 3.3

Templates >> Exercício 4

- Consulte o enunciado do exercício no material.

Macro Expansion TAL (METAL)

- Permite definir macros, isto é, fragmentos de código HTML e TAL que podem ser reutilizados.



METAL >> Macros

- *meutemplate.pt*
  -
    BLA

- *Outro template:*
  - <div metal:use-macro=
    "context/meutemplate/macros/minhamacro"/>

METAL >> Slots

- Dentro de uma macro podem ser definidos slots.
- São “buracos” onde o usuário da macro pode inserir um código personalizado:

```
<div metal:define-macro=“mostra_nome”>
   <b>Meu nome é:</b>
   <b metal:define-slot=“nome”>Um nome</b>
</div>
```

METAL >> Slots

- Se chamarmos a macro assim:

<div metal:use-macro
="context/teste/macros/mostra_nome">

- Aparecerá: Meu nome é: **Um nome**

METAL >> Slots

- Se chamarmos a macro assim:

```
<div metal:use-macro
   =“context/teste/macros/mostra_nome”>
  <b metal:fill-slot=”nome”>João</b>
</div>
```

- Aparecerá: Meu nome é: **João**

METAL Slots >> Exercício 5

- Módulo 3 > Exercício 5

METAL Slots >> Exercício 6

- Para criar um novo template já com o layout do portal, basta usar a macro “master”.
- Módulo 3 > Exercício 6

Perguntas?

Scripts Python no Plone

Scripts Python no Plone

- Scripts Python podem ser criados e invocados separadamente ou por templates.
- Servem para fazer processamentos simples.
  - Coisas mais complexas ficam em Produtos e bibliotecas (veremos mais à frente).
- Como tudo no Plone, são criados como objetos:
  - Acessar a ZMI.
  - Adicionar um novo objeto do tipo Script (Python).

Python Scripts >> Exercício 1

Obs: Criar o Script (Python) meuscript5

```
# Lista de pessoas
pessoas = [
    'Antonio Augusto',
    'Flavio Andreas',
    'Winstom Marsalis',
    'Chico Amaral',
    'Leo Brauer',
]

return pessoas
```

Python Scripts >> Exercício 2

```
# Lista de pessoas
pessoas = [
    'Antonio Augusto',
    'Flavio Andreas',
    'Winstom Marsalis',
    'Chico Amaral',
    'Leo Brauer',
]
for pessoa in pessoas:
    print pessoa
return printed
```

Python Scripts >> Exercício 3

```
# Lista de pessoas
pessoas = [ 'Antonio Augusto',   'Flavio Andreas', 'Winstom Marsalis', 'Chico Amaral',
'Leo Brauer',]
dados = { }
import random
for pessoa in pessoas:
    # Gera numeros aleatorios
    randomID = random.randint(1, 10000)
    randomIdade = random.randint(18,50)
    # Atribui a idade aleatoria e o ID aleatorio para a pessoa da lista
    dados[pessoa] = { 'idade':randomIdade, 'ID':randomID }
for pessoa in dados.keys():
    print pessoa + ' : ' + str(dados[pessoa])
return printed
```

Python Scripts >> Exercício 4

Life, the Universe, and Everything

A) Reescreva os números da entrada (uma lista). Pare de processar após ler o número 42. Retorne o resultado em uma lista.

Ex: entrada = [2,102, 42, 874, 900] -> saida = [2, 102, 42]

Obs: usar a entrada acima.

B) Fazer com que a lista entrada seja aleatória, gerando 10000 números entre 1 e 100 e colocando em uma lista.

Usando scripts em templates

```
<html metal:use-macro=
  ”context/main_template/macros/master">
  <div metal:fill-slot="main"
    tal:define="conteudo context/meu_script">
    <p tal:content="conteudo" />
  </div>
</html>
```

Exercício 5

- Confira o enunciado do exercício no material

Exercício 6

- Podemos interagir com as chamadas *portal tools* através de Scripts Python.
- Abordar o ZODB e portal_catalog.
- Confira o enunciado do exercício no material

Perguntas?