Templates

Os templates abaixo serão criados na pasta custom em portal_skins.

Exercício 1

Criar um template com ID meutemplate1, título Titulo do meutemplate1, e com o seguinte código:

```
<html>
    <head>
        <title tal:content="template/title">The title</title>
    </head>
    <body>

        Este é o template cujo título é:
        <div tal:replace="template/title"> </div>
        <br />
        Este template está sendo aplicado ao objeto:
        <div tal:replace="context/getId"> </div>
        <br />
        Você está na seguinte URL:
        <div tal:replace="request/URL"> </div>
        <br />
        Você está logado com o usuário:
        <div tal:replace="user/getId"> </div>
        <br />

    </body>
</html>
```

Chamar o template na raiz do site e a partir de algum objeto e ver o resultado.
Suponha que o portal se chame site1 e tenha um objeto chamado obj1 na raiz:

```
http://....site1/meutemplate1

http://....site1/obj1/meutemplate1
```

Troque um dos replace por content e observe o resultado.

Exercício 2

Crie um template chamado **template2**.

Altere o template para refletir os exemplos abaixo.

Estude as saídas de cada exemplo.

2.1) Soma Python

```
<h1> 2.1 Soma Python </h1>
<p>1+2 = <em tal:content="python:1+2"> </em></p>
```

2.2) Datas

```
<h1> 2.2 Datas </h1>
<span tal:define="data python:modules['DateTime'].DateTime()">
    <p>DateTime completo = <em tal:content="data" /></p>
    <p>DateTime formatado = <em tal:content="python:data.strftime('%d/%m/%y')" /></p>
    <p>DateTime extenso = <em tal:content="python: data.strftime('%A, %d de %B de %Y')"
/></p>
    <p>Hoje é dia de número <em tal:content="python: data.strftime('%j do ano.')" /></p>
</span>
```

2.3) Loop

```
<h1> 2.3 Loop </h1>
<span tal:define="lista python:[1,2,3,4,5,6,7,8,9,10]">
    <tal:loop repeat="obj lista">
        <span tal:content="obj"></span><br />
    </tal:loop>
</span>
```

2.4) Structure

```
<h1> 2.4 Structure </h1>
<p tal:replace="structure context/logo.png"> </p>
<br />
<p tal:replace="structure string:<strong>TEXTO EM NEGRITO</strong>"> </p>
```

2.5) Usando Structure nos dicionarios CONTEXTS (mostra as variáveis internas)

```
 <h1> 2.5 CONTEXTS </h1>
<h2>Debug information</h2>
 <h3>CONTEXTS</h3>
 <ul>
    <tal:block tal:repeat="item CONTEXTS">
        <li tal:condition="python: item != 'request'" tal:define="context CONTEXTS;">
            <b tal:content="item" />
            <span tal:replace="python: context[item]" />
        </li>
    </tal:block>
 </ul>
 <h3>REQUEST</h3>
 <p tal:replace="structure request" />
```

Exercício 3

Crie outro template chamado **meutemplate3** e teste os códigos a seguir.

3.1) Links para objetos

```
<h1> 3.1 Links para objetos </h1>
<b>Meu contexto é:</b>
<a tal:attributes="href context/absolute_url">
    <span tal:content="context/getId">Um Objeto</span>
</a>
```

3.2) tal:condition: Avaliando condicionais

```
<h1> 3.2 Avaliando condicionais </h1>
<div tal:define="messagem request/messagem | nothing">
 <p tal:condition="messagem">
A mensagem é <strong tal:content="messagem"> </strong>
 </p>
 <p tal:condition="not: messagem">
 Sem mensagem.
 </p>
</div>
```

Passar a mensagem na url:

```
http://localhost:8080/site1/meutemplate3?messagem=Qualquer mensagem
```

3.3) tal:repeat: Executando loops

```
<h1> 3.3 Executando loops </h1>

<table>
    <tr>
        <td> NÚMERO </td>
        <td> TÍTULO </td>
    </tr>
    <tr tal:repeat="linha context/portal_catalog">
        <td tal:content="repeat/linha/number">1</td>
        <td tal:content="linha/Title">Title</td>
    </tr>
</table>
```

Exercício 4: Desafio

Escrever um template chamado **meutemplate4** que atenda às seguintes especificações:

- Recuperar todos os objetos do portal através do portal_catalog.
- Definir uma variável que armazene todos os objetos do portal.
- Iterar sobre a lista de objetos recuperada.
- Mostrar a lista de objetos em forma de tabela.
- Os dados mostrados, por objeto, devem ser o id (atributo getId), o titulo e o tipo do objeto (atributo portal_type).

Exercício 5: Metal

Criar um template **meutemplate5_def** contendo a definição da seguinte macro:

```
<div metal:define-macro="mostra_nome">
    <span>Meu nome é:</span>
    <span metal:define-slot="nome">Um nome</span>
</div>
```

Criar outro template chamado **meutemplate5_use** contento as duas chamadas de macro seguintes:

```
<div metal:use-macro="context/meutemplate5_def/macros/mostra_nome"></div>

<br />
<br />

<div metal:use-macro="context/meutemplate5_def/macros/mostra_nome">
    <strong metal:fill-slot="nome">Pedro</strong>
</div>
```

Testar o template e observar o resultado das chamadas.

Exercício 6: Macro Master

Criar o template **meutemplate6** e testar o seguinte código:

```
<metal:block use-macro="context/main_template/macros/master">
    <metal:block fill-slot="main">
        Agora temos todo o layout do portal!
    </metal:block>
</metal:block>
```

Aplicar a macro master ao Exercício 4 (**meutemplate4**).

Dica visual 1: Utilize a classe listing (class="listing") na tabela.
Dica visual 2: No cabeçalho da tabela (primeira linha), utilize th no lugar do td.

Scripts Python no Plone

Os exercícios de 1 a 4 estão nos slides.

Exercício 5

Crie um template com o seguinte código:

```
<ol>
    <li tal:repeat="pessoa context/obter_pessoas" tal:content="pessoa">
        Uma pessoa
    </li>
</ol>
```

Crie um Script Python com o ID obter_pessoas e o conteúdo:

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

return pessoas
```

Teste o template e veja o resultado.

Exercício 6

1 - Criar um script na pasta custom em portal_skins.

2 - Adicionar o seguinte código:

```
from Products.CMFCore.utils import getToolByName

pc = getToolByName(context,"portal_catalog")
brains = pc.search({"portal_type":"Folder"})
for entry in brains:
    obj = entry.getObject()
    print obj

return printed
```

3 - Rodar e analisar a busca.