WebDynpro ABAP – Conteúdo MIME dinâmico no cache do AS

19/12/2012 10:00

Fala zumbizada esperta! Que título monstruoso, não?

Hoje estou passando aqui pra deixar uma técnica relacionada a WDA que eu "recolhi" de alguns sites e que se mostrou bem útil.

Quando você está trabalhando com **conteúdo MIME dinâmico** (i.e. arquivos de qualquer espécie), não dá pra usar o repositório do seu componente, porque obviamente você não tem o conteúdo nem o tipo dele em tempo de desenvolvimento (pense num job ou uma função que gera um binário de um PDF ou de uma imagem JPG de algum repositório).

Solução? Fazer o **upload no cache do servidor** onde está rodando a sua aplicação, e depois chamar a URL gerada para mostrar o conteúdo. Simples e tranquilo.

A única pegadinha é que não dá pra consultar diretamente a tabela do cache pra saber se o seu conteúdo ainda está válido ali, mas dá pra resolver isso usando o **tempo de expiração do objeto**. Vamos ao código:

```
DATA: lv_hora_expiracao TYPE timestamp,
      lv_comp               TYPE        i,
      lv_timestamp          TYPE        timestamp,
      lv_tp_conteudo        TYPE        string,
      lv_content_type       TYPE        string,
      lo_cached_response    TYPE REF TO if_http_response,
      lv_conteudo           TYPE        xstring,
      lv_url                TYPE        string,
      lv_nome_arquivo       TYPE        string,
      lv_expires_abs_date   TYPE        d,
      lv_expires_abs_time   TYPE        t,
      lv_guid               TYPE        guid_32.

* Antes de começar o processo, verificamos se o conteúdo que já foi
* colocado anteriormente no cache ainda está válido. Se ainda estiver
* válido, não é preciso fazer um novo upload.
IF NOT lv_hora_expiracao IS INITIAL.

  GET TIME STAMP FIELD lv_timestamp.

  lv_comp = cl_abap_tstmp=>compare(
      tstmp1 = lv_timestamp
      tstmp2 = lv_hora_expiracao
  ).

ENDIF.

IF lv_comp <> -1.
* Neste caso, o timestamp atual é igual ou superior ao timestamp de
* expiração, portanto temos que subir o conteúdo novamente para o
* cache.

************************************************************************
* 1. Determinação do conteúdo
************************************************************************
* Criamos primeiro o objeto do cache:
  CREATE OBJECT lo_cached_response
    TYPE cl_http_response
    EXPORTING
      add_c_msg = 1.
```

```
* O conteúdo aqui vem de alguma fonte dinâmica (upload, módulo de
* função, gravado em alguma tabela, etc.) em formato XSTRING:
  lo_cached_response->set_data( lv_conteudo ).

* Determinamos o MIME type do conteúdo que vai ser gravado. Abaixo
* estão 2 tipos mais comuns, adapte conforme a sua necessidade:
  CASE lv_tp_conteudo.
    WHEN 'JPG'.
      lv_content_type = 'image/jpeg'.
    WHEN 'PDF'.
      lv_content_type = 'application/pdf'.
    WHEN OTHERS.
*     ...
  ENDCASE.

  lo_cached_response->set_header_field(
      name  = if_http_header_fields=>content_type
      value = lv_content_type
  ).

* Determinando o status do pacote (sempre OK):
  lo_cached_response->set_status(
      code   = 200
      reason = 'OK'
  ).

*********************************************************************
* 2. Determinando a URL
*********************************************************************
* Normalmente é possível usar a própria URL da sua aplicação, que pode
* ser obtida com a classe CL_WD_UTILITIES. Caso isso não seja possível,
* verifique com o Basis um caminho.

  cl_wd_utilities=>construct_wd_url(
    EXPORTING
      application_name = wd_this->wd_get_api( )->get_application( )->get_application_info( )->get_
    IMPORTING
      out_local_url    = lv_url
  ).

* Se você tiver o nome do arquivo, use o prório junto com a URL.
  lv_url = lv_url && '/' && lv_nome_arquivo.

* Caso você não tenha ou o conteúdo seja dinâmico, crie um caminho
* usando a função GUID_CREATE:
  CALL FUNCTION 'GUID_CREATE'
    IMPORTING
      ev_guid_32 = lv_guid.

  lv_url = lv_url && '/' && lv_guid. "não precisa de extensão

*********************************************************************
* 3. Definindo a validade do conteúdo
*********************************************************************
* Essa validade tem que ser gravada no contexto junto da URL, para que
* possamos testar depois

  GET TIME STAMP FIELD lv_hora_expiracao.

* Qualquer valor entre 1 e 3min é razoável, porém ajuste conforme o
* seu ambiente
  lv_hora_expiracao = cl_abap_tstmp=>add(
     tstmp = lv_hora_expiracao
     secs  = 180
  ).
```

```
  CONVERT TIME STAMP lv_hora_expiracao
    TIME ZONE 'BRAZIL'
    INTO DATE lv_expires_abs_date
         TIME lv_expires_abs_time.

  lo_cached_response->server_cache_expire_abs(
    EXPORTING
      expires_abs_date = lv_expires_abs_date
      expires_abs_time = lv_expires_abs_time
  ).

**********************************************************************
* 4. Upload do conteúdo
**********************************************************************
  cl_http_server=>server_cache_upload(
      url      = lv_url
      response = lo_cached_response
  ).

ENDIF.
```

Se você tiver aquele aceso IXXXPERTO você pode conferir o cache na transação SMICM indo em *Goto > HTTP Plug-In > Server Cache > Display*.

Por enquanto é isso, espero que seja útil para vocês. E eu juro que vou terminar a série de segurança (só que esse ano não dá mais né?)

Comentários

> **Felipe Simionatto** — *15/09/2013 14:48*
>
> Muito útil seu post, vlw. Para não limitar a extensão dos arquivos no código, você pode utilizar a função SDOK_MIMETYPE_GET para pegar o mime type.

> > **Leo Schmidt** — *15/09/2013 22:47*
> >
> > Ótima dica, valeu!

> **Fawcs** — *20/12/2012 09:48*
>
> esse ano a gente não tem mais segurança no sap=(