ColdFusion

Coding Guidelines

Fevereiro 2005

1 Introdução....2
2 Nomenclatura, estilo e comentários....2
2.1 Nomenclatura....2
2.1.1 Convenções gerais de nomenclatura....2
2.1.2 Abreviações....2
2.1.3 Nomes de arquivos....3
2.1.4 Tags, atributos e operadores....3
2.1.5 Campos, funções, propriedades e variáveis....3
2.1.6 Valores de atributos....4
2.2 Comentários....4
2.2.1 Convenções gerais....4
2.2.2 Comentários de arquivos....4
2.2.3 Comentários em Componentes....5
2.3 Layout....5
2.3.1 Formato do arquivo....5
2.3.2 HTML e XHTML....5
2.3.3 CFML e XHTML....6
2.3.4 Código SQL....6
3 Bancos de dados....7
3.1 Convenções gerais....7
4 Apêndice: Melhores práticas e performance....7
4.1 Convenções gerais....7
4.1.1 Use....7
4.1.2 Não use....8
4.1.3 Evite....8
4.2 Funções....8
4.2.1 Argumentos....8
4.3 Componentes....9
4.3.1 Construtores....9
4.4 Query....9
5 Histórico do documento....9

1 Introdução

Este documento visa estabelecer padrões de codificação para aplicações ColdFusion. Estes padrão foram adotados com base no ColdFusion MX 6.1 e 7.0.

Este documento pode sofrer atualizações constantes, de acordo com novas versões do ColdFusion Server e convenções internas adotadas e aprovadas.

Este documento foi primariamente baseado na experiência pessoal e no ColdFusion MX Coding Guidelines (v 3.1, 12/25/2004), da Macromedia.

2 Nomenclatura, estilo e comentários

Esta seção prove convenções de nomenclatura (arquivos, tags, variáveis, etc) e estilo de formatação de código.

2.1 Nomenclatura

Esta seção provê convenções de nomenclatura de elementos do código ColdFusion.

2.1.1 Convenções gerais de nomenclatura

Duas premissas são fundamentais nessa convenção: consistência e facilidade de leitura.

- Deve-se utilizar palavras em inglês para descrever o elementos;
- Em geral, usa-se “verbo_substantivo” e “adjetivo_substantivo”;
- Em geral, usa-se o substantivo no singular. O uso no plural é permitido quando a ação envolver mais de um elemento.
    - getEmployeeName()
    - list_employees.cfm
- caixaMista é um estilo utilizado em alguns elementos, e tem a inicial minúscula, com as iniciais das demais palavras em maíscula. Também conhecida como *camelCaseHeadLess*.

Bons exemplos:

```
calculate_tax.cfm
userServices.cfc
lista_usuarios.cfm
userName
userPhoto
getUserData
dsp_form_user.cfm
umNomeMuitoCompridoEFacilDeLer
um_nome_muito_comprido_e_facil_de_ler
```

Maus exemplos:

```
umnomemuitocompridoedificildeler
usuarios_lista.cfm
```

2.1.2 Abreviações

Uso somente de abreviações padrão, como ID, URL e COD. Outras abreviações não são convenções e não se apoiam em facilidade de leitura e entendimento.

2.1.3 Nomes de arquivos

- Não devem conter espaços;
- Sempre em minúscula, separando as palavras por underscore;
    - Exceção para nomes de arquivos de componentes (CFCs) utilizam *TitleCase*, com as iniciais de cada palavra em maiusculo, como no padrão utilizado pelo Java.
    - Exceção para Application.cfm, Application.cfc e OnRequestEnd.cfm por se tratar de um padrão do ColdFusion Server.
- Sufixos:
    - Páginas HTML: .html
    - Páginas CFML: .cfm
    - Arquivos de Componentes ColdFusion: .cfc
    - Documentos XML: .xml
    - Documentos CSS: .css

2.1.4 Tags, atributos e operadores

- Todas as tags (CFML e HTML) em caixa baixa
- Atributos das tags em caixa baixa;
- Operadores em caixa alta;
- Não utilize espaços entre o atributo e o valor caso o valor seja literal:
    - Bom exemplo: <table border=”1”>
    - Mau exemplo: <table border = “1”>
- Utilize espaço entre o atributo e o valor caso o valor seja uma expressão:
    - Bom exemplo: <cfset variables.myVar = application.myVar />

Exemplo:

```
<cfif campo1 NEQ campo2>
      <cfquery name=”nome” datasorce=”dsn”>
      …
```

2.1.5 Campos, funções, propriedades e variáveis

- Estes elementos utilizam *caixaMista*
- Prefixos utilizados em variáveis:
    - Boleanas: *is* ou *has*. Ex.: isAdmin, hasParent
        - Único tipo de dado simples com prefixo.
    - Arrays: *a*. Ex.: aNames
    - XML: *x*. Ex.: xPackages
    - Structure: *st*. Ex.: stJobs
    - WDDX: *w*. Ex.: wPackages
    - Binário: *bin*. Ex.: binUserPhoto
    - Query: *q*. Ex.: qListUsers
        - Todas as queries, independente se selecionam, atualizam, inserem ou deletam dados, devem conter o atributo nome corretamente preenchido.
- Variáveis devem ser referenciadas utilizando seu escopo.
    - O escopo deve estar em letra minúscula: Ex: *application.name*, url.*ID*
    - A exceção é em variáveis locais de funções. Ex.: <cfset var i=0>

2.1.6 Valores de atributos

- Os valores, exceto da tag cfreturn, devem estar entre aspas dupla
    - Na tag cfset, o uso de aspas dupla é permitido apenas se o valor atribuido for literal, isto é, não for uma variável.
        - <cfset nome="Fabio">
        - <cfset nome=myVariable>
- Aspas simples é permitido utilizar no caso do valor já conter aspas dupla
- Não use nomes de variáveis processadas (*evaluated*), como *stStruct.#variable#.* Utilize *stStruct[variable].*

2.2 Comentários

2.2.1 Convenções gerais

Utilize comentários CFML <!-- … ---> para todos os elementos importantes do código, descrevendo o que o código faz, e porquê, descrevendo também como, se não for óbvio.

Ao realizar uma alteração, comente-a. Identifique a data e seu nome de usuário:

```
<!--- 2005-03-29 fabio.terracini CHANGE: Alterado comentário --->
```

Para deixar uma nota sobre um bug ou uma funcionalidade e implementar, use a palavra-chave TODO:

```
<!--- 2005-03-29 fabio.terracini TODO: Finalizar o documento --->
```

Tais palavras-chave, além de ajudar outros desenvolvedores, pode ser lido por IDEs como o Eclipse e montar uma lista de tarefas automaticamente.

2.2.2 Comentários de arquivos

Todo arquivo CFML deve começar com um comentários contendo o nome do arquivo, uma mensagem padrão de copyright, seguida de uma explicação do que o arquivos faz, bem como um cabeçalho padrão do CVS ($Id: $).

```
<!---
Esta página mostra para o usuário o pacote correspondente ao ID
passado. Essa página é chamada através do fuse packages.show. Este
fuse também inclui uma query, que é a que pega os dados do banco.

@cvs        $Id: dsp_package.cfm,v 1.7 2005/03/29 21:46:27
fabio.terracini

@author     Fabio Terracini
@created    2005-03-31
@copyright  (c) 2005 Empresa S/A

@param      url.packageID - O ID, em valor numérico, do pacote a
            ser mostrado ao usuário
@see        /index.cfm
@see        qry_package.cfm

--->
```

O comentário deve conter, obrigatoriamente, os seguintes elementos:

- Descrição: uma breve descrição do que o arquivo faz, inclusive para componentes e custom tags
- @cvs: o código para atualização automática pelo CVS ($Id: $)
- @author: o autor da versão inicial do arquivo
- @created: data de criação do arquivo, em formato japonês: yyyy-mm-dd
- @copyright: uma mensagem padrão de copyright

E o comentário pode conter também os seguintes elementos:

- @param: variáveis de entrada com uma breve descrição. Não necessário em caso de componentes
- @result: variáveis de saída com uma breve descrição. Não necessário em caso de componentes
- @see: arquivos do que arquivo o atual depende, e quais arquivos dependem do atual

2.2.3 Comentários em Componentes

- As tags *cfcomponent*, *cfproperty*, *cffunction* e *cfargument* têm o atributo *displayName* e *hint* que devem ser preenchidos em todos os casos
    - Utilize uma descrição menos técnica (como para um usuário de negócio) no atributo *displayName*
    - Detalhes mais técnicos e/ou forma de utilização no atributo *hint*.
- Para a tag *cffunction*, se a função gerar um erro, o atributo *hint* deve conter: *
Throws: documentar o erro que a função pode gerar* no final da hint.

2.3 Layout

Esta seção provê convenções do layout do código ColdFusion e HTML.

2.3.1 Formato do arquivo

- O arquivo deve ser salvo em CRLF
- O código deve ser identado
- Toda identação deve ser feita com tabs de 4 espaços (não espaços em si, mas o próprio tab).

2.3.2 HTML e XHTML

Todo HTML gerado deve ser compatível com XHTML:

- Caixa baixa em todos os elementos e atributos;
- Todos os valores utilizando corretamente as aspas;
- Fechar todas as tags corretamente
    - e
- Fechar tags sem par
    -
- Documentos compatíveis com XHTML devem iniciar com o seguinte código:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

2.3.3 CFML e XHTML

- O código ColdFusion não pode ser escrito puramente compatível com XHTML, devido a certas tags (*cfif*, *cfelse*, *cfreturn* e *cfset*), mas deve ser feito um esforço para tornar o código mais compatível com XHTML possível.
    - Colocar aspas em valores literais (<cfset myVar=”123”>)
    - Caixa baixa em tags e atributos
    - Fechar os elementos possíveis com a tag par.
        - A excessão é com as tags cfreturn, cfset e cfelse, que não tem par.

- Para evitar problemas com Custom Tags que não tem par (evitar que o auto-fechamento chame a custom tag duas vezes), todas as custom tags devem verificar o *executionMode*:

```
<cfswitch expression="#thisTag.executionMode#">
   <cfcase value="start">
   ...
   </cfcase>
   <cfcase value="end">
   ...
   </cfcase>
</cfswitch>
```

2.3.4 Código SQL

O código SQL deve ser formatada utilizando quebras de linha de acordo com o exemplo abaixo:

Exemplo 1:

```
SELECT
    TO.COLUMN_ONE,
    TO.COLUMN_TWO,
    TT.COLUMN_THREE
FROM
    TABLE_ONDE TO,
    TABLE_TEW TT
WHERE
    TO.TABLE_ONE_ID = TT.TABLE_TWO_ID
    AND TT.TABLE_TWO_ID = 10
ORDER BY
    TO.TABLE_ONE_ORDER_KEY
```

Exemplo 2:

```
INSERT INTO
    TABLE_ONDE
(
    COLUMN_ONDE,
    COLUMN_TWO,
    COLUMN_THREE
)
VALUES
(
    'ValueOne',
    'ValueTwo',
    'ValueThree'
)
```

Exemplo 3:

```
UPDATE
      TABLE_ONE
SET
    COLUMN_ONE = 'ValueOne',
    COLUMN_TWO = 'ValueTwo'
WHERE
    TABLE_ONE_ID = 10
    AND COLUMN_THREE = 'ValueThree'
```

Exemplo 4:

```
DELETE FROM
      TABLE_ONE
WHERE
      TABLE_ONE_ID = 10
```

3 Bancos de dados

Esta seção provê convenções de nomenclatura e estrutura em banco de dados.

3.1 Convenções gerais

- Nomes de tabelas em geral devem estar no plural, utilizando caixa alta, com as palavras separadas por underscore
    - CUSTOMERS
    - APP_USERS
- Os nomes de tabelas devem evitar conflitos com tabelas padrão em outros banco de dados, como por exemplo, não utilizar tabelas que comecem com *sys*, já que o SQL Server utiliza tabelas com esse nome, como *sysusers*.
- Nomes de colunas devem estar no singular, utilizando caixa alta, com as palavras separadas por underscore
    - USER_NAME
    - CUSTOMER_ID
    - LABEL
- Evitar repetir o nome da tabela no nome da coluna
- Evitar abreviações, exceto as padrões (ID, COD, etc)
- Chaves primárias devem ter o sufixo *ID*
    - Exemplo: USER_ID, CUSTOMER_ID
- Em geral, todas as tabelas devem conter *DATE_CREATED* e DATE_LAST_UPDATED
- Caso o banco seja de terceiros, utiliza a nomenclatura e caixa utilizada.

4 Apêndice: Melhores práticas e performance

4.1 Convenções gerais

4.1.1 Use

- Use arrays ao invés de listas, mas não converte listas para arrays para fazer poucas ou pequenas comparações.
    - Ganhos de performance

- Use a conversão automática de expressões para valores boleanos
    - Errado: <cfif isAdmin IS “true”>
    - Errado: <cfif userName NEQ “”>
    - Certo: <cfif isAdmin>
    - Certo: <cfif len(userName)>

4.1.2 Não use

- Não use # desnecessariamente
    - Errado: <cfset variable=”#anotherVariable#”>
    - Certo: <cfset variable=anotherVariable>
- Não use incrementValue()
    - Prefira a notação x = x + 1
    - Além de ser mais legível, é mais rápida

4.1.3 Evite

- Evite utilizar evaluate()
- Evite utilizar iif()
    - Um bloco de cfif/cfelse é mais rápido e mais legível
- Evite cfelseif para uma grande quantidade de condições
    - Prefira cfswitch/cfcase
    - Nos blocos de cfswitch, coloque no topo as cases mais utilizadas
- Evite o uso de cfmodule
    - É mais devagar que uma chamada a um componente, e até mesmo mais devagar que uma chamada a uma custom tag.
    - Prefira fazer a chamada como customTag
    - Ou utilize a tag cfimport

4.2 Funções

- Sempre inicialize as variáveis da função no topo do código, logo abaixo dos argumentos, utilizando a palavra *var*
    - Exemplo: <cfset var i = 0>
    - Não esquecer também de inicializar certas variáveis, como as utilizadas em cfloop, cfquery e as variáveis de retorno (seja em cfinvoke ou dentro de uma função).
- Prefira criar e utilizar funções que utilizem a notação em tag (não em cfscript).

4.2.1 Argumentos

Embora o ColdFusion não exiga o uso da tag cfargument em funções, esta sempre deve ser utilizada, já que provê validação para a entrada de dados e funciona como uma documentação adicional da função.

- Sempre especifique o atributo type. Evite utilizar type=”any”
- Sempre especifique o atributo required
- Se o argumento for requerido, não especifique o atributo default.
    - No caso do argumento não ser requerido, em geral especifique o atributo default.
    - Caso seja necessário verficar se um argument não requerido foi passado, não especifique o valor default e use structKeyExists(arguments.”argumentName”) na função.
- Estas regras não se aplicam caso a função esteja em *cfscript*.

4.3 Componentes

4.3.1 Construtores

- Todo componente deve ter um método init() que inicialize a instância, mesmo que o corpo da função esteja vazio.
    - Exemplo para o componente com.empresa.projeto.util
      <cffunction name="init" returntype="com.empresa.projeto.util">
        <cfreturn this>
      </cffunction>
- Se o componente extender outro component, o método init deste deve primeiro fazer uma chamada a super.init() para inicializar o componente pai.

4.4 Query

- Sempre utilize cfqueryparam
    - Além de oferecer ganhos de performance, também visa a segurança
- Queries de insert, delete e update devem retornar *true* (returntype="boolean" portanto) ou um erro (com *cfthrow),* que será tratado pela aplicação, tanto em Flash/Flex quanto em ColdFusion.
- Utilize o atributo blockfactor
    - Faça uma estimativa de quantas linhas do resultado da query cabem em 32k, que é o buffer do ColdFusion em conexão à banco de dados
    - Por exemplo: um campo de texto longo, uma data e um titulo.
        - Texto longo: varchar(5000), data: datetime(8), titulo: varchar(100): máximo de 5108 bytes
        - 32768 bytes / 5108 bytes ~= 6 linhas por vez
        - blockfactor="6"

5 Histórico do documento

Data	Autor	Versão
03/Fev/05	Fabio Terracini	Versão 1: Versão inicial do documento.