Documentação do GUtils

Contents

Package GUtils Procedural Elements

GExtenso.php

GExtenso class file

- **Package** GUtils
- **Author** Fausto Gonçalves Cintra (goncin) < goncin@gmail.com>
- **Link** http://twitter.com/g0nc1n
- **Link** http://devfranca.ning.com
- **Filesource** Source Code for this file
- **License** http://creativecommons.org/licenses/LGPL/2.1/deed.pt

Package GUtils Classes

Class GExtenso

[line 50]

GExtenso é uma classe que gera a representação por extenso de um número ou valor monetário.

Sua implementação foi feita como prova de conceito, utilizando:

- Métodos estáticos, implementando o padrão de projeto (*design pattern*) **SINGLETON**;
- Chamadas recursivas a métodos, minimizando repetições e mantendo o código enxuto;
- Uso de pseudoconstantes ('private static') diante das limitações das constantes de classe;
- Tratamento de erros por intermédio de exceções; e
- Utilização do phpDocumentor (http://www.phpdoc.org) para documentação do código fonte e geração automática de documentação externa.

EXEMPLOS DE USO

Para obter o extenso de um número, utilize GExtenso:: numero.

```
echo GExtenso::numero(832); // oitocentos e trinta e dois
echo GExtenso::numero(832, GExtenso::GENERO_FEM) // oitocentas e trinta e duas
```

Para obter o extenso de um valor monetário, utilize GExtenso:: moeda.

```
// IMPORTANTE: veja nota sobre o parâmetro 'valor' na documentação do método!
echo GExtenso::moeda(15402); // cento e cinquenta e quatro reais e dois centavos
echo GExtenso::moeda(47); // quarenta e sete centavos
echo GExtenso::moeda(357082, 2,
  array('peseta', 'pesetas', GExtenso::GENERO_FEM),
  array('cêntimo', 'cêntimos', GExtenso::GENERO_MASC));
  // três mil, quinhentas e setenta pesetas e oitenta e dois cêntimos
```

- **Package** GUtils

- **Author** Fausto Gonçalves Cintra (goncin) < goncin@gmail.com>
- **Version** 0.1 2010-03-02

GExtenso::GENERO_FEM

= 1 *[line 56]*

GExtenso::GENERO_MASC

= 0 *[line 55]*

GExtenso::NUM_PLURAL

= 1 *[line 53]*

GExtenso::NUM_SING

= 0 *[line 52]*

GExtenso::POS_GENERO

= 2 *[line 54]*

GExtenso::VALOR_MAXIMO

= 999999999 *[line 58]*

string function GExtenso::moeda($valor, [$casasDecimais = 2], [$infoUnidade = array('real', 'reais', self::GENERO_MASC)], [$infoFracao = array('centavo', 'centavos', self::GENERO_MASC)]) *[line 285]*

Function Parameters:

- *int* **$valor** O valor monetário cujo extenso se deseja gerar. ATENÇÃO: PARA EVITAR OS CONHECIDOS PROBLEMAS DE ARREDONDAMENTO COM NÚMEROS DE PONTO FLUTUANTE, O VALOR DEVE SER PASSADO JÁ DEVIDAMENTE MULTIPLICADO POR 10 ELEVADO A $casasDecimais (o que equivale, normalmente, a passar o valor com centavos multiplicado por 100)
- *int* **$casasDecimais** (Opcional; valor padrão: 2) Número de casas decimais a serem consideradas como parte fracionária (centavos)
- *array* **$infoUnidade** (Opcional; valor padrão: array('real', 'reais', GExtenso::GENERO_MASC)) Fornece informações sobre a moeda a ser utilizada. O primeiro valor da matriz corresponde ao nome da moeda no singular, o segundo ao nome da moeda no plural e o terceiro ao gênero gramatical do nome da moeda (GExtenso::GENERO_MASC ou GExtenso::GENERO_FEM)
- *array* **$infoFracao** (Opcional; valor padrão: array('centavo', 'centavos', self::GENERO_MASC)) Provê informações sobre a parte fracionária da moeda. O primeiro valor da matriz corresponde ao nome da parte fracionária no singular, o segundo ao nome da parte fracionária no plural e o terceiro ao gênero gramatical da parte fracionária (GExtenso::GENERO_MASC ou GExtenso::GENERO_FEM)

Gera a representação por extenso de um valor monetário, maior que zero e menor ou igual a GExtenso::VALOR_MAXIMO.

- **Since** 0.1 2010-03-02
- **Static**
- **Access** public

string function GExtenso::numero($valor, [$genero = self::GENERO_MASC]) *[line 167]*

Function Parameters:

- *int* **$valor** O valor numérico cujo extenso se deseja gerar
- *int* **$genero** (Opcional; valor padrão: GExtenso::GENERO_MASC) O gênero gramatical (GExtenso::GENERO_MASC ou GExtenso::GENERO_FEM) do extenso a ser gerado. Isso possibilita distinguir, por exemplo, entre 'duzentos e dois homens' e 'duzentas e duas mulheres'.

Gera a representação por extenso de um número inteiro, maior que zero e menor ou igual a GExtenso::VALOR_MAXIMO.

- **Since** 0.1 2010-03-02
- **Static**
- **Access** public

Appendices

Appendix A - Class Trees

Package GUtils

GExtenso

- GExtenso

Appendix C - Source Code

Package GUtils

File Source for GExtenso.php

Documentation for this file is available at GExtenso.php

```
1     <?php
2     /**
3      * GExtenso class file
4      *
5      * @author Fausto Gonçalves Cintra (goncin) <goncin@gmail.com>
6      * @link http://devfranca.ning.com
7      * @link http://twitter.com/g0nc1n
8      * @license http://creativecommons.org/licenses/LGPL/2.1/deed.pt
9      */
10
11    /**
12     * GExtenso é uma classe que gera a representação por extenso de um número ou valor monetário.
13     *
14     * Sua implementação foi feita como prova de conceito, utilizando:
15     *
16     * <ul>
17     * <li>Métodos estáticos, implementando o padrão de projeto (<i>design
pattern</i>) <b>SINGLETON</b>;</li>
18     * <li>Chamadas recursivas a métodos, minimizando repetições e mantendo o código
enxuto;</li>
19     * <li>Uso de pseudoconstantes ('private static') diante das limitações das constantes de
classe;</li>
20     * <li>Tratamento de erros por intermédio de exceções; e</li>
21     * <li>Utilização do phpDocumentor (     {@link http://www.phpdoc.org}) para documentação do código
fonte e
22     * geração automática de documentação externa.</li>
23     * </ul>
24     *
25     * <b>EXEMPLOS DE USO</b>
26     *
27     * Para obter o extenso de um número, utilize GExtenso::{@link numero}.
28     * <pre>
29     * echo GExtenso::numero(832); // oitocentos e trinta e dois
30     * echo GExtenso::numero(832, GExtenso::GENERO_FEM) // oitocentas e trinta e duas
31     * </pre>
32     *
33     * Para obter o extenso de um valor monetário, utilize GExtenso::{@link moeda}.
34     * <pre>
35     * // IMPORTANTE: veja nota sobre o parâmetro 'valor' na documentação do método!
36     * echo GExtenso::moeda(15402); // cento e cinquenta e quatro reais e dois centavos
37     * echo GExtenso::moeda(47); // quarenta e sete centavos
38     * echo GExtenso::moeda(357082, 2,
39     *   array('peseta', 'pesetas', GExtenso::GENERO_FEM),
40     *   array('cêntimo', 'cêntimos', GExtenso::GENERO_MASC));
41     *   // três mil, quinhentas e setenta pesetas e oitenta e dois cêntimos
42     * </pre>
43     *
44     * @author Fausto Gonçalves Cintra (goncin) <goncin@gmail.com>
45     * @version 0.1 2010-03-02
46     * @package GUtils
47     *
48     */
49
50     class GExtenso {
51
52      const NUM_SING = 0;
53      const NUM_PLURAL = 1;
54      const POS_GENERO = 2;
55      const GENERO_MASC = 0;
56      const GENERO_FEM = 1;
57
58      const VALOR_MAXIMO = 999999999;
59
60      /* Uma vez que o PHP não suporta constantes de classe na forma de matriz (array),
61        a saída encontrada foi declarar as strings numéricas como 'private static'.
62      */
63
```

```
64      /* As unidades 1 e 2 variam em gênero, pelo que precisamos de dois conjuntos de strings
(masculinas e femininas) para as unidades */
65      private static $UNIDADES = array(
66        self::GENERO_MASC => array(
67          1 =>    'um',
68          2 =>    'dois',
69          3 =>    'três',
70          4 =>    'quatro',
71          5 =>    'cinco',
72          6 =>    'seis',
73          7 =>    'sete',
74          8 =>    'oito',
75          9 =>    'nove'
76        ),
77        self::GENERO_FEM => array(
78          1 =>    'uma',
79          2 =>    'duas',
80          3 =>    'três',
81          4 =>    'quatro',
82          5 =>    'cinco',
83          6 =>    'seis',
84          7 =>    'sete',
85          8 =>    'oito',
86          9 =>    'nove'
87        )
88      );
89
90      private static $DE11A19 = array(
91        11 =>    'onze',
92        12 =>    'doze',
93        13 =>    'treze',
94        14 =>    'quatorze',
95        15 =>    'quinze',
96        16 =>    'dezesseis',
97        17 =>    'dezessete',
98        18 =>    'dezoito',
99        19 =>    'dezenove'
100     );
101
102     private static $DEZENAS = array(
103       10 =>    'dez',
104       20 =>    'vinte',
105       30 =>    'trinta',
106       40 =>    'quarenta',
107       50 =>    'cinquenta',
108       60 =>    'sessenta',
109       70 =>    'setenta',
110       80 =>    'oitenta',
111       90 =>    'noventa'
112     );
113
114     private static $CENTENA_EXATA = 'cem';
115
116     /* As centenas, com exceção de 'cento', também variam em gênero. Aqui também se faz
117       necessário dois conjuntos de strings (masculinas e femininas).
118     */
119
120     private static $CENTENAS = array(
121       self::GENERO_MASC => array(
122         100 =>    'cento',
123         200 =>    'duzentos',
124         300 =>    'trezentos',
125         400 =>    'quatrocentos',
126         500 =>    'quinhentos',
127         600 =>    'seiscentos',
128         700 =>    'setecentos',
129         800 =>    'oitocentos',
130         900 =>    'novecentos'
131       ),
132       self::GENERO_FEM => array(
133         100 =>    'cento',
134         200 =>    'duzentas',
135         300 =>    'trezentas',
136         400 =>    'quatrocentas',
137         500 =>    'quinhentas',
138         600 =>    'seiscentas',
139         700 =>    'setecentas',
140         800 =>    'oitocentas',
141         900 =>    'novecentas'
142       )
```

```
143     );
144
145     /* 'Mil' é invariável, seja em gênero, seja em número */
146     private static $MILHAR = 'mil';
147
148     private static $MILHOES = array(
149       self::NUM_SING =>    'milhão',
150       self::NUM_PLURAL =>    'milhões'
151     );
152
153
154
155    /**
156     * Gera a representação por extenso de um número inteiro, maior que zero e menor ou igual a
GExtenso::VALOR_MAXIMO.
157     *
158     * @param int O valor numérico cujo extenso se deseja gerar
159     *
160     * @param int (Opcional; valor padrão: GExtenso::GENERO_MASC) O gênero gramatical
(GExtenso::GENERO_MASC ou GExtenso::GENERO_FEM)
161     *  do extenso a ser gerado. Isso possibilita distinguir, por exemplo, entre 'duzentos e dois
homens' e 'duzentas e duas mulheres'.
162     *
163     * @return string O número por extenso
164     *
165     * @since 0.1 2010-03-02
166     */
167     public static function numero($valor, $genero = self::GENERO_MASC) {
168
169       /* ----- VALIDAÇÃO DOS PARÂMETROS DE ENTRADA ---- */
170
171       if(!is_numeric($valor))
172         throw new Exception("     [Exceção em GExtenso::numero] Parâmetro \$valor não é numérico
(recebido: '$valor')"     );
173
174       else if($valor <=    0)
175         throw new Exception("     [Exceção em GExtenso::numero] Parâmetro \$valor igual a ou menor que
zero (recebido: '$valor')"     );
176
177       else if($valor >    self::VALOR_MAXIMO)
178         throw new Exception('[Exceção em GExtenso::numero] Parâmetro $valor deve ser um inteiro entre 1
e ' . self::VALOR_MAXIMO . "      (recebido: '$valor')"     );
179
180       else if($genero != self::GENERO_MASC &&         $genero != self::GENERO_FEM)
181         throw new Exception("     Exceção em GExtenso: valor incorreto para o parâmetro \$genero
(recebido: '$genero')."     );
182
183       /* ------------------------------------------------ */
184
185       else if($valor >=    1 &&         $valor <=    9)
186         return self::$UNIDADES[$genero][$valor]; // As unidades 'um' e 'dois' variam segundo o gênero
187
188       else if($valor == 10)
189         return self::$DEZENAS[$valor];
190
191       else if($valor >=    11 &&         $valor <=    19)
192         return self::$DE11A19[$valor];
193
194       else if($valor >=    20 &&         $valor <=    99) {
195         $dezena = $valor - ($valor % 10);
196         $ret = self::$DEZENAS[$dezena];
197         /* Chamada recursiva à função para processar $resto se este for maior que zero.
198          * O conectivo 'e' é utilizado entre dezenas e unidades.
199          */
200         if($resto = $valor - $dezena) $ret .= ' e ' . self::numero($resto, $genero);
201         return $ret;
202       }
203
204       else if($valor == 100) {
205         return self::$CENTENA_EXATA;
206       }
207
208       else if($valor >=    101 &&         $valor <=    999) {
209         $centena = $valor - ($valor % 100);
210         $ret = self::$CENTENAS[$genero][$centena]; // As centenas (exceto 'cento') variam em gênero
211         /* Chamada recursiva à função para processar $resto se este for maior que zero.
212          * O conectivo 'e' é utilizado entre centenas e dezenas.
213          */
214         if($resto = $valor - $centena) $ret .= ' e ' . self::numero($resto, $genero);
215         return $ret;
```

```
216       }
217
218       else if($valor >=    1000 &&         $valor <=    999999) {
219         /* A função 'floor' é utilizada para encontrar o inteiro da divisão de $valor por 1000,
220          * assim determinando a quantidade de milhares. O resultado é enviado a uma chamada recursiva
221          * da função. A palavra 'mil' não se flexiona.
222          */
223         $milhar = floor($valor / 1000);
224         $ret = self::numero($milhar, self::GENERO_MASC) . ' ' . self::$MILHAR; // 'Mil' é do gênero
masculino
225         $resto = $valor % 1000;
226         /* Chamada recursiva à função para processar $resto se este for maior que zero.
227          * O conectivo 'e' é utilizado entre milhares e números entre 1 e 99, bem como antes de
centenas exatas.
228          */
229         if($resto && ((        $resto >=    1 &&         $resto <=    99) || $resto % 100 == 0))
230           $ret .= ' e ' . self::numero($resto, $genero);
231         /* Nos demais casos, após o milhar é utilizada a vírgula. */
232         else if ($resto)
233           $ret .= ', ' . self::numero($resto, $genero);
234         return $ret;
235       }
236
237       else if($valor >=    100000 &&         $valor <=    self::VALOR_MAXIMO) {
238         /* A função 'floor' é utilizada para encontrar o inteiro da divisão de $valor por 1000000,
239          * assim determinando a quantidade de milhões. O resultado é enviado a uma chamada recursiva
240          * da função. A palavra 'milhão' flexiona-se no plural.
241          */
242         $milhoes = floor($valor / 1000000);
243         $ret = self::numero($milhoes, self::GENERO_MASC) . ' '; // Milhão e milhões são do gênero
masculino
244
245         /* Se a o número de milhões for maior que 1, deve-se utilizar a forma flexionada no plural */
246         $ret .= $milhoes == 1 ? self::$MILHOES[self::NUM_SING] : self::$MILHOES[self::NUM_PLURAL];
247
248         $resto = $valor % 1000000;
249
250         /* Chamada recursiva à função para processar $resto se este for maior que zero.
251          * O conectivo 'e' é utilizado entre milhões e números entre 1 e 99, bem como antes de
centenas exatas.
252          */
253         if($resto && ((        $resto >=    1 &&         $resto <=    99) || $resto % 100 == 0))
254           $ret .= ' e ' . self::numero($resto, $genero);
255         /* Nos demais casos, após o milhão é utilizada a vírgula. */
256         else if ($resto)
257           $ret .= ', ' . self::numero($resto, $genero);
258         return $ret;
259       }
260
261     }
262
263     /**
264      * Gera a representação por extenso de um valor monetário, maior que zero e menor ou igual a
GExtenso::VALOR_MAXIMO.
265      *
266      * @param int O valor monetário cujo extenso se deseja gerar.
267      *  ATENÇÃO: PARA EVITAR OS CONHECIDOS PROBLEMAS DE ARREDONDAMENTO COM NÚMEROS DE PONTO FLUTUANTE, O
VALOR DEVE SER PASSADO
268      *  JÁ DEVIDAMENTE MULTIPLICADO POR 10 ELEVADO A $casasDecimais (o que equivale, normalmente, a
passar o valor com centavos
269      *  multiplicado por 100)
270      *
271      * @param int (Opcional; valor padrão: 2) Número de casas decimais a serem consideradas como parte
fracionária (centavos)
272      *
273      * @param array (Opcional; valor padrão: array('real', 'reais', GExtenso::GENERO_MASC)) Fornece
informações sobre a moeda a ser
274      *  utilizada. O primeiro valor da matriz corresponde ao nome da moeda no singular, o segundo ao
nome da moeda no plural e o terceiro
275      *  ao gênero gramatical do nome da moeda (GExtenso::GENERO_MASC ou GExtenso::GENERO_FEM)
276      *
277      * @param array (Opcional; valor padrão: array('centavo', 'centavos', self::GENERO_MASC)) Provê
informações sobre a parte fracionária
278      *  da moeda. O primeiro valor da matriz corresponde ao nome da parte fracionária no singular, o
segundo ao nome da parte fracionária no plural
279      *  e o terceiro ao gênero gramatical da parte fracionária (GExtenso::GENERO_MASC ou
GExtenso::GENERO_FEM)
280      *
281      * @return string O valor monetário por extenso
282      *
```

```
283     * @since 0.1 2010-03-02
284     */
285     public static function moeda(
286       $valor,
287       $casasDecimais = 2,
288       $infoUnidade = array('real', 'reais', self::GENERO_MASC),
289       $infoFracao = array('centavo', 'centavos', self::GENERO_MASC)
290     ) {
291
292       /* ----- VALIDAÇÃO DOS PARÂMETROS DE ENTRADA ---- */
293
294       if(!is_numeric($valor))
295         throw new Exception("     [Exceção em GExtenso::moeda] Parâmetro \$valor não é numérico
(recebido: '$valor')"     );
296
297       else if($valor <=    0)
298         throw new Exception("     [Exceção em GExtenso::moeda] Parâmetro \$valor igual a ou menor que
zero (recebido: '$valor')"     );
299
300       else if(!is_numeric($casasDecimais) || $casasDecimais <    0)
301         throw new Exception("     [Exceção em GExtenso::moeda] Parâmetro \$casasDecimais não é numérico
ou é menor que zero (recebido: '$casasDecimais')"     );
302
303       else if(!is_array($infoUnidade) || count($infoUnidade) <    3) {
304         $infoUnidade = print_r($infoUnidade, true);
305         throw new Exception("     [Exceção em GExtenso::moeda] Parâmetro \$infoUnidade não é uma matriz
com 3 (três) elementos (recebido: '$infoUnidade')"     );
306       }
307
308       else if($infoUnidade[self::POS_GENERO] != self::GENERO_MASC &&
$infoUnidade[self::POS_GENERO] != self::GENERO_FEM)
309         throw new Exception("     Exceção em GExtenso: valor incorreto para o parâmetro
\$infoUnidade[self::POS_GENERO] (recebido: '{$infoUnidade[self::POS_GENERO]}')."     );
310
311       else if(!is_array($infoFracao) || count($infoFracao) <    3) {
312         $infoFracao = print_r($infoFracao, true);
313         throw new Exception("     [Exceção em GExtenso::moeda] Parâmetro \$infoFracao não é uma matriz
com 3 (três) elementos (recebido: '$infoFracao')"     );
314       }
315
316       else if($infoFracao[self::POS_GENERO] != self::GENERO_MASC &&
$infoFracao[self::POS_GENERO] != self::GENERO_FEM)
317         throw new Exception("     Exceção em GExtenso: valor incorreto para o parâmetro
\$infoFracao[self::POS_GENERO] (recebido: '{$infoFracao[self::POS_GENERO]}')."     );
318
319       /* ------------------------------------------------ */
320
321       /* A parte inteira do valor monetário corresponde ao $valor passado dividido por 10 elevado a
$casasDecimais, desprezado o resto.
322        * Assim, com o padrão de 2 $casasDecimais, o $valor será dividido por 100 (10^2), e o resto é
descartado utilizando-se floor().
323        */
324       $parteInteira = floor($valor / pow(10, $casasDecimais));
325
326       /* A parte fracionária ('centavos'), por seu turno, corresponderá ao resto da divisão do $valor
por 10 elevado a $casasDecimais.
327        * No cenário comum em que trabalhamos com 2 $casasDecimais, será o resto da divisão do $valor
por 100 (10^2).
328        */
329       $fracao = $valor % pow(10, $casasDecimais);
330
331       /* O extenso para a $parteInteira somente será gerado se esta for maior que zero. Para tanto,
utilizamos
332        * os préstimos do método GExtenso::numero().
333        */
334       if($parteInteira) {
335         $ret = self::numero($parteInteira, $infoUnidade[self::POS_GENERO]) . ' ';
336         $ret .= $parteInteira == 1 ? $infoUnidade[self::NUM_SING] : $infoUnidade[self::NUM_PLURAL];
337       }
338
339       /* De forma semelhante, o extenso da $fracao somente será gerado se esta for maior que zero. */
340       if($fracao) {
341         /* Se a $parteInteira for maior que zero, o extenso para ela já terá sido gerado. Antes de
juntar os
342          * centavos, precisamos colocar o conectivo 'e'.
343          */
344         if ($parteInteira) $ret .= ' e ';
345         $ret .= self::numero($fracao, $infoFracao[self::POS_GENERO]) . ' ';
346         $ret .= $parteInteira == 1 ? $infoFracao[self::NUM_SING] : $infoFracao[self::NUM_PLURAL];
347       }
```

```
348
349         return $ret;
350
351       }
352
353     }
354     ?>
```

Index

G