عنوان: **تبدیل عدد به حروف**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۶/۲۸ ۱۵:۰۵:۰۰
آدرس: www.dotnettips.info
گروه‌ها: Tips

به طور قطع توابع و کلاس‌های تبدیل عدد به حروف، در جعبه ابزار توابع کمکی شما هم پیدا می‌شوند. روز قبل سعی کردم جهت آزمایش، عدد 3000,000,000,000,000 ریال را با کلاسی که دارم تست کنم و نتیجه overflow یا اصطلاحا ترکیدن سیستم بود! البته اگر مطالب این سایت را دنبال کرده باشید پیشتر در همین راستا مطلبی در مورد نحوه‌ی صحیح بکارگیری توابع تجمعی SQL در این سایت منتشر شده است و جزو الزامات هر سیستمی است (تفاوتی هم نمی‌کند که به چه زبانی تهیه شده باشد). اگر آن‌را رعایت نکرده‌اید، سیستم شما «روزی» دچار overflow خواهد شد.

در کل این کلاس تبدیل عدد به حروف را به صورت ذیل اصلاح کردم و همچنین دو زبانه است؛ چیزی که کمتر در پیاده سازی‌های عمومی به آن توجه شده است:

```
using System.Collections.Generic;
using System.Linq;

namespace NumberToWordsLib
{
 /// <summary>
 /// Number to word languages
 /// </summary>
 public enum Language
 {
  /// <summary>
  /// English Language
  /// </summary>
  English,

  /// <summary>
  /// Persian Language
  /// </summary>
  Persian
 }

 /// <summary>
 /// Digit's groups
 /// </summary>
 public enum DigitGroup
 {
  /// <summary>
  /// Ones group
  /// </summary>
  Ones,

  /// <summary>
  /// Teens group
  /// </summary>
  Teens,

  /// <summary>
  /// Tens group
  /// </summary>
  Tens,

  /// <summary>
  /// Hundreds group
  /// </summary>
  Hundreds,

  /// <summary>
  /// Thousands group
  /// </summary>
  Thousands
 }

 /// <summary>
```

```
 /// Equivalent names of a group
 /// </summary>
 public class NumberWord
 {
  /// <summary>
  /// Digit's group
  /// </summary>
  public DigitGroup Group { set; get; }

  /// <summary>
  /// Number to word language
  /// </summary>
  public Language Language { set; get; }

  /// <summary>
  /// Equivalent names
  /// </summary>
  public IList<string> Names { set; get; }
 }

 /// <summary>
 /// Convert a number into words
 /// </summary>
 public static class HumanReadableInteger
 {
  #region Fields (4)

  private static readonly IDictionary<Language, string> And = new Dictionary<Language, string>
  {
   { Language.English, " " },
   { Language.Persian, " و " }
  };
  private static readonly IList<NumberWord> NumberWords = new List<NumberWord>
  {
   new NumberWord { Group= DigitGroup.Ones, Language= Language.English, Names=
    new List<string> { string.Empty, "One", "Two", "Three", "Four", "Five", "Six", "Seven", "Eight",
"Nine" }},
   new NumberWord { Group= DigitGroup.Ones, Language= Language.Persian, Names=
    new List<string> { string.Empty, "نه", "هشت", "هفت", "شش", "پنج", "چهار", "سه", "دو", "یک"
}},

   new NumberWord { Group= DigitGroup.Teens, Language= Language.English, Names=
    new List<string> { "Ten", "Eleven", "Twelve", "Thirteen", "Fourteen", "Fifteen", "Sixteen",
"Seventeen", "Eighteen", "Nineteen" }},
   new NumberWord { Group= DigitGroup.Teens, Language= Language.Persian, Names=
    new List<string> { "","سیزده", "چهارده", "پانزده", "شانزده", "هفده", "هجده", "نوزده"
"" }},

   new NumberWord { Group= DigitGroup.Tens, Language= Language.English, Names=
    new List<string> { "Twenty", "Thirty", "Forty", "Fifty", "Sixty", "Seventy", "Eighty", "Ninety" }},
   new NumberWord { Group= DigitGroup.Tens, Language= Language.Persian, Names=
    new List<string> { "نود", "هشتاد", "هفتاد", "شصت", "پنجاه", "چهل", "سی", "بیست" }},

   new NumberWord { Group= DigitGroup.Hundreds, Language= Language.English, Names=
    new List<string> {string.Empty, "One Hundred", "Two Hundred", "Three Hundred", "Four Hundred",
     "Five Hundred", "Six Hundred", "Seven Hundred", "Eight Hundred", "Nine Hundred" }},
   new NumberWord { Group= DigitGroup.Hundreds, Language= Language.Persian, Names=
    new List<string> {string.Empty, "","هفتصد", "ششصد", "پانصد", "چهارصد", "سیصد", "دویست", "یکصد"
"هشتصد"" }},

   new NumberWord { Group= DigitGroup.Thousands, Language= Language.English, Names=
     new List<string> { string.Empty, " Thousand", " Million", " Billion"," Trillion", " Quadrillion",
" Quintillion", " Sextillian",
   " Septillion", " Octillion", " Nonillion", " Decillion", " Undecillion", " Duodecillion", "
Tredecillion",
   " Quattuordecillion", " Quindecillion", " Sexdecillion", " Septendecillion", " Octodecillion", "
Novemdecillion",
   " Vigintillion", " Unvigintillion", " Duovigintillion", " 10^72", " 10^75", " 10^78", " 10^81", "
10^84", " 10^87",
   " Vigintinonillion", " 10^93", " 10^96", " Duotrigintillion", " Trestrigintillion" }},
   new NumberWord { Group= DigitGroup.Thousands, Language= Language.Persian, Names=
     new List<string> { string.Empty, " تریلیون", " میلیارد", " میلیون", " هزار", " Quadrillion", "
Quintillion", " Sextillian",
   " Septillion", " Octillion", " Nonillion", " Decillion", " Undecillion", " Duodecillion", "
Tredecillion",
   " Quattuordecillion", " Quindecillion", " Sexdecillion", " Septendecillion", " Octodecillion", "
Novemdecillion",
   " Vigintillion", " Unvigintillion", " Duovigintillion", " 10^72", " 10^75", " 10^78", " 10^81", "
10^84", " 10^87",
   " Vigintinonillion", " 10^93", " 10^96", " Duotrigintillion", " Trestrigintillion" }},
  };
```

```
private static readonly IDictionary<Language, string> Negative = new Dictionary<Language, string>
{
 { Language.English, "Negative " },
 { Language.Persian, "منهای " }
};
private static readonly IDictionary<Language, string> Zero = new Dictionary<Language, string>
{
 { Language.English, "Zero" },
 { Language.Persian, "صفر" }
};

#endregion Fields

#region Methods (7)

// Public Methods (5)

/// <summary>
/// display a numeric value using the equivalent text
/// </summary>
/// <param name="number">input number</param>
/// <param name="language">local language</param>
/// <returns>the equivalent text</returns>
public static string NumberToText(this int number, Language language)
{
 return NumberToText((long)number, language);
}


/// <summary>
/// display a numeric value using the equivalent text
/// </summary>
/// <param name="number">input number</param>
/// <param name="language">local language</param>
/// <returns>the equivalent text</returns>
public static string NumberToText(this uint number, Language language)
{
 return NumberToText((long)number, language);
}

/// <summary>
/// display a numeric value using the equivalent text
/// </summary>
/// <param name="number">input number</param>
/// <param name="language">local language</param>
/// <returns>the equivalent text</returns>
public static string NumberToText(this byte number, Language language)
{
 return NumberToText((long)number, language);
}

/// <summary>
/// display a numeric value using the equivalent text
/// </summary>
/// <param name="number">input number</param>
/// <param name="language">local language</param>
/// <returns>the equivalent text</returns>
public static string NumberToText(this decimal number, Language language)
{
 return NumberToText((long)number, language);
}

/// <summary>
/// display a numeric value using the equivalent text
/// </summary>
/// <param name="number">input number</param>
/// <param name="language">local language</param>
/// <returns>the equivalent text</returns>
public static string NumberToText(this double number, Language language)
{
 return NumberToText((long)number, language);
}

/// <summary>
/// display a numeric value using the equivalent text
/// </summary>
/// <param name="number">input number</param>
/// <param name="language">local language</param>
/// <returns>the equivalent text</returns>
public static string NumberToText(this long number, Language language)
{
```

```
   if (number == 0)
   {
    return Zero[language];
   }

   if (number < 0)
   {
    return Negative[language] + NumberToText(-number, language);
   }

   return wordify(number, language, string.Empty, 0);
  }
  // Private Methods (2)

  private static string getName(int idx, Language language, DigitGroup group)
  {
   return NumberWords.Where(x => x.Group == group && x.Language == language).First().Names[idx];
  }

  private static string wordify(long number, Language language, string leftDigitsText, int thousands)
  {
   if (number == 0)
   {
    return leftDigitsText;
   }

   var wordValue = leftDigitsText;
   if (wordValue.Length > 0)
   {
    wordValue += And[language];
   }

   if (number < 10)
   {
    wordValue += getName((int)number, language, DigitGroup.Ones);
   }
   else if (number < 20)
   {
    wordValue += getName((int)(number - 10), language, DigitGroup.Teens);
   }
   else if (number < 100)
   {
    wordValue += wordify(number % 10, language, getName((int)(number / 10 - 2), language,
DigitGroup.Tens), 0);
   }
   else if (number < 1000)
   {
    wordValue += wordify(number % 100, language, getName((int)(number / 100), language,
DigitGroup.Hundreds), 0);
   }
   else
   {
    wordValue += wordify(number % 1000, language, wordify(number / 1000, language, string.Empty,
thousands + 1), 0);
   }

   if (number % 1000 == 0) return wordValue;
   return wordValue + getName(thousands, language, DigitGroup.Thousands);
  }

  #endregion Methods
 }
}
```

دریافت پروژه کامل به همراه Unit tests مرتبط

نظرات خوانندگان

نویسنده: Farhad Yazdan-Panah
تاریخ: ۱۳۹۰/۰۶/۲۸ ۱۸:۵۲:۲۸

بسیار طراحی جالبی داشت.
البته به نظر من اگر اون ثوابت رو (معادل ها) از یک فایل (XML یا ...) بخونه تا حدودی بهتر میشه (برای زبان جدید).
البته جسارت نشه.فقط کاش برای اعداد ممیز شناور (Floating point) هم یه فکری می کردید.چون گاهی لازم میشه اعداد (امتیاز و ...) رو به حروف نوشت.

نویسنده: Farhad Yazdan-Panah
تاریخ: ۱۳۹۰/۰۶/۲۸ ۱۹:۵۳:۳۶

گویا موتور گزارشگیری Stimulsoft در نسخه 2011.2 یک تابع مشابه به موضوع رو به کنابانه خودش اضافه کرده است.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۶/۲۸ ۲۳:۱۲:۱۲

چند بحث کلی اینجا هست:
- چون عموما از عدد به حروف در گزارشات مالی استفاده می‌شود، اعداد همه int و big int هستند. بنابراین آنچنان کاربرد دنیای واقعی ندارد سایر حالت‌ها.
- بحث اضافه کردن سایر زبان‌ها ... خوب، بستگی به تسلط به زبان‌های مختلف هم دارد. مثلا در انگلیسی می‌گویند Three hundred اما در فارسی مرسوم نیست که کسی بگه «سه صد». به همین جهت یک قسمت اضافه‌تر برای معرفی سیصد و امثال آن در کد فوق وجود دارد. به احتمال زیاد زبان‌های دیگر هم ریزه‌کاری‌های خاص خودشان را دارند.
- بحث سرعت را هم در نظر بگیرید. در این نوع الگوریتم‌ها به علت استفاده مکرر، ترجیح داده می‌شود که از کالکشن‌های تشکیل شده در حافظه (بجای خواندن از فایل) جهت سرعت بالاتر دسترسی به اطلاعات و سربار کمتر استفاده شود.

نویسنده: Farhad Yazdan-Panah
تاریخ: ۱۳۹۰/۰۶/۲۹ ۰۰:۳۳:۱۱

ممنون از توضیحاتتون.
- در مورد کاربرد بیشتر تبدیل اعداد صحیح موافقم، ولی در سیستم های آکادمیک (حیطه کاری من)، همانند: سیستم های پژوهشی، آموزشی، و ... معمولا حالت ممیز شناور نیز دارای کاربردهای زیادیه. (امتیاز، گرنت، معدل، و ...)- در مورد سایر زبان ها حق با شماست و هر زبونی برای خودش یه داستانی داره!!
- منظور من از خواندن از فایل بیشتر برای مقدار دهی اولیه بود (استفاده از الگوی singleton) و جدا کردن دادگان از الگوریتم بود. چون بعیده که کسی بخواد مقادیر رو در زمان اجرا عوض کنه. مسلما سرعت در درجه بالایی از اهمیته و مقیم بودن همیشگی در حافظه یک راه حل مناسبه.