TextFileGenerator

author: Iuga Alexandru (lastunicorn)  
version: 1.0.1  
  
date: 2015 04 30

# overview

## why does it exist?

from time to time i was needing huge txt files with dummy content for testing some algorithms. i was tired to create each time a small tool from scratch so i created a configurable tool to use it when i need. that’s how the TextFileGenerator was born.

## what can it do?

this tool can create huge text files with dummy data generated based on a description provided in an xml file.

currently the application has only a console interface. i have plans for a gui, but nothing sure until now.

# using the console ui

the console reads a descriptor file (xml), based on which it knows how to generate the resulted text file.



## create the descriptor file

the descriptor file looks like this:

<?xml version="1.0"?>

<textFileGenerator xmlns="http://alez.ro/TextFileGenerator">

<sections>

...

</sections>

</textFileGenerator>

you can copy-paste this scaffold from this document or you can generate it using the -x option:

c:\> TextFileGenerator.exe -x

this command will generate the file file.xml in the current directory.

## generate the huge text file

the console receives, as a sole parameter, an xml file that describes the text file to be generated.

c:\> TextFileGenerator.exe text1.xml

---------------------------------------------

creates in the same directory the text1.output.txt file

# the descriptor file (input)

## the <section> tag

the generated text file is assembled as a collection of sections. each section is described by a <section> tag in the xml file.

### attributes

* name [optional] - identifies the section - currently not used
* repeat [optional] - an integer greater than zero that specifies how many times the section to be rendered.
* separator [optional] - a text that separates the instances of the section.
* separatorLocation [optional] - Prefix, Infix, Postfix - specifies where to use the separator text. [need more explanation]

<section> ... </section>

<section repeat="10" separator=";" separatorLocation="Infix"> ... </section>

example of section repeated 10 times:

<section repeat="5" separator=";" separatorLocation="Infix">

<text>text</text>

</section>

----------------------------------------------

text;text;text;text;text

obs: no semicolon at the end because the “Infix” value.

### content

* <text> tag - the text that the section will output. only one <text> tag can be added. if this tag is added there can be no <section> tags.

<section>

<text>some text here</text>

</section>

----------------------------------------------

some text here

* <section> tags - list of other sections. there can be multiple <section> tags. if at least one <section> tag is added there can be no <text> tag.

<section>

<section>

<text>section1</text>

</section>

<section>

<text>section2</text>

</section>

</section>

----------------------------------------------

section1section2

* <parameter> tags - can exists multiple instances of this tag. should be added after the <text> or <section> tags. they are used inside the <text> or <section> tags.

## the <parameter> tag

the parameters are calculated values that are inserted in the text of a section.

the valuePersistence attribute specifies when the parameter value is changed:

* PerRequest - no persistence at all. the value of the parameter changes each time somebody requests it. if the parameter is requested multiple times by the template, it will return different values each time.
* PerSectionStep - the value of the parameter remains constant for the rendering of the current iteration of the section. if the parameter is requested multiple times by the template, it will return the same value each time.

### counter

starts from a number and increments it for each iteration of the section.

<counter/> // 1, 2, 3 ...

<counter format="000" startValue="10" step=”2”/> // 10, 12, 14, 16 ...

the format parameter is a c# format string.

### randomNumber

generates a random number between min and max for each iteration of the section.

<randomNumber/> // random in interval [1, 100]

<randomNumber format=”000” minValue=”100” maxValue=”200”/> // random in interval [100, 200]

the format parameter is a c# format string.

### randomText

generates a text with random characters for each iteration of the section.

available chars: “abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890”.

<randomText/> // random text between 1 and 100 characters.

<randomText minLength=”5” maxLength=”10”/> // random text between 5 and 10 characters.

example of section with random text:

<section>

<text>text plus one parameter: {param\_name}</text>

<parameter name="param\_name">

<randomText/>

</parameter>

</section>

----------------------------------------------

text plus one parameter: rVyLL0VaGuSb0QI3luWArhvPfcLAVtroSKc7t

### constant

not really useful. it is a static text. its value can be easily put into the section’s text itself.

<constant/> // <empty string>

<constant value="some text"/> // some text

# examples

## how to create a csv file with 3 lines and 5 columns

<section name="row" repeat="3" separator="&#13;&#10;" separatorLocation="Infix">

<section name="cell" repeat="5" separator="," separatorLocation="Infix">

<text>text\_{rowIndex}\_{cellIndex}</text>

<parameter key="cellIndex" valuePersistence="PerSectionStep">

<counter/>

</parameter>

</section>

<parameter key="rowIndex" valuePersistence="PerSectionStep">

<counter/>

</parameter>

</section>

----------------------------------------------

text\_1\_1,text\_1\_2,text\_1\_3,text\_1\_4,text\_1\_5

text\_2\_1,text\_2\_2,text\_2\_3,text\_2\_4,text\_2\_5

text\_3\_1,text\_3\_2,text\_3\_3,text\_3\_4,text\_3\_5

obs: no crlf at the end of file because the first “Infix” value.

obs: no semicolon at the end of line because the second “Infix” value.

# in the future

* add attribute: parameter->@valueProvider that can be counter, randomNumber, randomText, constant.

in the future - valueProvider attribute for parameter tag:

<section name="row" repeat="3" separator="&#13;&#10;" separatorLocation="Infix">

<section name="cell" repeat="5" separator="," separatorLocation="Infix">

<text>text\_{rowIndex}\_{cellIndex}</text>

<parameter name="cellIndex" valueProvider="counter"/>

</section>

<parameter name="rowIndex" valueProvider="counter"/>

</section>

----------------------------------------------

text\_1\_1,text\_1\_2,text\_1\_3,text\_1\_4,text\_1\_5

text\_2\_1,text\_2\_2,text\_2\_3,text\_2\_4,text\_2\_5

text\_3\_1,text\_3\_2,text\_3\_3,text\_3\_4,text\_3\_5

in the future - a whole section as parameter:

<section name="cell" repeat="4" separator="," separatorLocation="Infix">

<text>text\_{aaa}</text>

<parameter name="aaa">

<section repeat="3" separator="|" separatorLocation="Infix">aaa</section>

</parameter>

</section>

----------------------------------------------

text\_aaa|aaa|aaa,text\_aaa|aaa|aaa,text\_aaa|aaa|aaa,text\_aaa|aaa|aaa

in the future: additional parameter that takes two other parameters and switches between them.

in the future: parameter->@valuePersistence:

* Eternal - the value of the parameter is requested once from the value provider and then it is not change. the value will remain the same in all the iterations of the section.