Documentation
Projet gus05

Titre : Transformations textuelles pour Gus Text Tool

Auteur : Augustin Delale

Version : 13/12/2009

Transformations textuelles
pour Gus Text Tool

Après avoir défini la transformation textuelle come type d'entité gus05, ce document présente un ensemble de transformations textuelles simples qui seront utilisées dans le cadre du développement de l'outil de traitement de texte "Gus Text Tool". Chaque entité est présentée associée à une icône de visualisation, son code source et une petite description de la transformation. Par ailleurs, une application de test est fournit afin de vous permettre de tester l'ensemble des entités présentées sur des textes de votre choix.

Définition d'une transformation textuelle

Une transformation textuelle est un type d'entité défini de la manière suivante :

1. Il s'agit d'une entité de multiplicité UNIQUE
2. Cette entité est dotée de la caractéristique Transform
3. La caractéristique Transform reçoit en entrée un objet String et renvoie en sortie un autre objet String

On peut de plus préciser la relation sémantique suivante :

1. L'objet en sortie de la caractéristique Transform est considérée comme le résultat de la transformation de l'objet en entrée.

Signature du type : Tra(String,String)+

Différentes catégories de transformations textuelles

On peut distinguer plusieurs catégories de transformation textuelle suivant les aspects du textes qui sont modifiés par la transformation. Voici les catégories qui ont été définies :

1. Transformations qui agissent sur chaque caractère du texte
    1. Transformations qui modifient la case des caractères du texte
    2. Transformations qui modifient l'ordre des caractères du texte
2. Transformations qui agissent sur chaque ligne du texte
    1. Transformations qui modifient l'ordre des lignes du texte
    2. Transformations qui suppriment des lignes du texte
    3. Transformations qui ajoutent une indication devant chaque ligne du texte
    4. Transformations qui font des calculs statistiques sur les lignes du texte

Récapitulatif de toutes les entités

Entité	Description
Transformations qui agissent sur les caractéres du texte	
Transformations qui modifient la case des caractères du texte	
gus.stringtransform.character.uppercase	met en majuscule les caractères du texte
gus.stringtransform.character.lowercase	met en minuscule les caractères du texte
gus.stringtransform.character.invertcase	inverse la case des caractères du texte
gus.stringtransform.character.randomcase	met aléatoirement en majuscule ou en minuscule chaque caractère du texte
gus.stringtransform.character.wordlowercase	met en minuscule la première lettre de chaque mot du texte
gus.stringtransform.character.worduppercase	met en majuscule la première lettre de chaque mot du texte
Transformations qui modifient l'ordre des caractères du texte	
gus.stringtransform.character.sort	trie des caractères du texte
gus.stringtransform.character.invert	inverse l'ordre des caractères du texte
gus.stringtransform.character.shuffle	change aléatoirement l'ordre des caractères du texte
Transformations qui agissent sur les lignes du texte	
Transformations qui modifient l'ordre des lignes du texte	
gus.stringtransform.line.sort	trie les lignes du texte par ordre alphabétique
gus.stringtransform.line.sortlength	trie les lignes du texte selon leurs longueurs
gus.stringtransform.line.invert	inverse l'ordre des lignes du texte
gus.stringtransform.line.shuffle	change aléatoirement l'ordre des lignes du texte
Transformations qui suppriment des lignes du texte	
gus.stringtransform.line.empty	supprime toutes les lignes vides (ou blanches) du texte
gus.stringtransform.line.doublon	supprime toutes les lignes précédemment apparues dans le texte
gus.stringtransform.line.regroup	réduit les lignes identiques contiguës du texte à une seule ligne
Transformations qui ajoutent une indication devant chaque ligne du texte	
gus.stringtransform.line.index	fait commencer chaque ligne du texte par le numéro de ligne et une tabulation
gus.stringtransform.line.length	fait commencer chaque ligne du texte par sa longueur et une tabulation
gus.stringtransform.line.occ	fait commencer chaque ligne du texte par son nombre d'occurrences et une tabulation
Transformations qui font des calculs statistiques sur les lignes du texte	

F gus.stringtransform.line.frequency	affiche les fréquences de chaque ligne trouvée dans le texte
J gus.stringtransform.line.repartitionfactor	affiche les facteurs de répartition de chaque ligne trouvée dans le texte
H gus.stringtransform.line.repartitionlisting	affiche les répartitions de chaque ligne trouvée dans le texte
G gus.stringtransform.line.averageposition	affiche les positions moyennes de chaque ligne trouvée dans le texte
B gus.stringtransform.line.buildtab	

gus.stringtransform.character.uppercase

Transformation textuelle qui met en majuscule les caractères du texte.

```
package gus05.entity.gus.stringtransform.character.uppercase;

import gus05.framework.core.Entity;
import gus05.framework.features.Transform;

public class TransformCharUppercase implements Entity, Transform
{
    public String getName()      {return "gus.stringtransform.character.uppercase";}
    public String getCreationDate(){return "2006.06.21";}

    public Object transform(Object obj) throws Exception
    {
       String text = (String) obj;
       return text.toUpperCase();
    }
}
```

gus.stringtransform.character.lowercase

Transformation textuelle qui met en minuscule les caractères du texte.

```
package gus05.entity.gus.stringtransform.character.lowercase;

import gus05.framework.core.Entity;
import gus05.framework.features.Transform;

public class TransformCharLowercase implements Entity, Transform
{
    public String getName()      {return "gus.stringtransform.character.lowercase";}
    public String getCreationDate(){return "2006.06.21";}

    public Object transform(Object obj) throws Exception
    {
       String text = (String) obj;
       return text.toLowerCase();
    }
}
```

gus.stringtransform.character.invertcase

Transformation textuelle qui inverse la case des caractères du texte.

```
package gus05.entity.gus.stringtransform.character.invertcase;

import gus05.framework.core.Entity;
import gus05.framework.features.Transform;

public class TransformCharInvertCase implements Entity, Transform
{
    public String getName()      {return "gus.stringtransform.character.invertcase";}
    public String getCreationDate(){return "2006.06.21";}
```

```
    public Object transform(Object obj) throws Exception
    {
        String text = (String) obj;

        StringBuffer b = new StringBuffer();
        for(int i=0;i<text.length();i++)
        {
            char c = text.charAt(i);
            if(Character.isUpperCase(c))
                b.append(Character.toLowerCase(c));
            else b.append(Character.toUpperCase(c));
        }
        return b.toString();
    }
}
```

gus.stringtransform.character.randomcase

Transformation textuelle qui met aléatoirement en majuscule ou en minuscule chaque caractère du texte.

```
package gus05.entity.gus.stringtransform.character.randomcase;

import gus05.framework.core.Entity;
import gus05.framework.features.Transform;

public class TransformCharRandomCase implements Entity, Transform
{
    public String getName()      {return "gus.stringtransform.character.randomcase";}
    public String getCreationDate(){return "2009.11.18";}

    public Object transform(Object obj) throws Exception
    {
        String text = obj.toString();

        StringBuffer b = new StringBuffer();
        for(int i=0;i<text.length();i++)
        {
            char c = text.charAt(i);
            if(Math.random()<0.5)
                b.append(Character.toLowerCase(c));
            else b.append(Character.toUpperCase(c));
        }
        return b.toString();
    }
}
```

gus.stringtransform.character.wordlowercase

Transformation textuelle qui met en minuscule la première lettre de chaque mot du texte.

```
package gus05.entity.gus.stringtransform.character.wordlowercase;

import gus05.framework.core.Entity;
import gus05.framework.features.Transform;

public class TransformCharWordLowercase implements Entity, Transform
{
    public String getName()      {return "gus.stringtransform.character.wordlowercase";}
    public String getCreationDate(){return "2006.06.21";}
```

```
    public Object transform(Object obj) throws Exception
    {
       String text = (String) obj;
        StringBuffer b = new StringBuffer();

        char c = ' ';
        for(int i=0;i<text.length();i++)
        {
            if(!Character.isLetterOrDigit(c))
                b.append(Character.toLowerCase(text.charAt(i)));
            else b.append(text.charAt(i));
            c = text.charAt(i);
        }
        return b.toString();
    }
}
```

gus.stringtransform.character.worduppercase

Transformation textuelle qui met en majuscule la première lettre de chaque mot du texte.

```
package gus05.entity.gus.stringtransform.character.worduppercase;

import gus05.framework.core.Entity;
import gus05.framework.features.Transform;

public class TransformCharWordUppercase implements Entity, Transform
{
    public String getName()      {return "gus.stringtransform.character.worduppercase";}
    public String getCreationDate(){return "2006.06.21";}

    public Object transform(Object obj) throws Exception
    {
        String text = (String) obj;
        StringBuffer b = new StringBuffer();

        char c = ' ';
        for(int i=0;i<text.length();i++)
        {
            if(!Character.isLetterOrDigit(c))
                b.append(Character.toUpperCase(text.charAt(i)));
            else b.append(text.charAt(i));
            c = text.charAt(i);
        }
        return b.toString();
    }
}
```

gus.stringtransform.character.sort

Transformation textuelle qui trie des caractères du texte.

```
package gus05.entity.gus.stringtransform.character.sort;

import java.util.Arrays;
import gus05.framework.core.Entity;
import gus05.framework.features.Transform;

public class TransformCharSort implements Entity, Transform
```

```
{
    public String getName()      {return "gus.stringtransform.character.sort";}
    public String getCreationDate(){return "2009.12.11";}

    public Object transform(Object obj) throws Exception
    {
       String text = (String) obj;
       char[] tab = text.toCharArray();
       Arrays.sort(tab);
       return new String(tab);
    }
}
```

gus.stringtransform.character.invert

Transformation textuelle qui inverse l'ordre des caractères du texte.

```
package gus05.entity.gus.stringtransform.character.invert;

import gus05.framework.core.Entity;
import gus05.framework.features.Transform;

public class TransformCharInvert implements Entity, Transform
{
    public String getName()      {return "gus.stringtransform.character.invert";}
    public String getCreationDate(){return "2006.06.21";}

    public Object transform(Object obj) throws Exception
    {
       StringBuffer b = new StringBuffer((String)obj);
        return b.reverse().toString();
    }
}
```

gus.stringtransform.character.shuffle

Transformation textuelle qui change aléatoirement l'ordre des caractères du texte.

```
package gus05.entity.gus.stringtransform.character.shuffle;

import gus05.framework.core.Entity;
import gus05.framework.features.Transform;

public class TransformCharShuffle implements Entity, Transform
{
    public String getName()      {return "gus.stringtransform.character.shuffle";}
    public String getCreationDate(){return "2006.06.21";}

    public Object transform(Object obj) throws Exception
    {
        StringBuffer in = new StringBuffer((String)obj);
        StringBuffer b = new StringBuffer();
        while(in.length()>0)
        {
            int n = (int)(Math.random()*in.length());
            b.append(in.charAt(n));
            in.deleteCharAt(n);
        }
        return b.toString();
    }
```

```
}
```

gus.stringtransform.line.sort

Transformation textuelle qui trie les lignes du texte par ordre alphabétique.

```
package gus05.entity.gus.stringtransform.line.sort;

import java.util.Arrays;
import gus05.framework.core.Entity;
import gus05.framework.features.Transform;

public class TransformLineSort implements Entity, Transform
{
    public String getName()      {return "gus.stringtransform.line.sort";}
    public String getCreationDate(){return "2006.06.21";}

    public Object transform(Object obj) throws Exception
    {
        String[] lines = obj.toString().split("[\n\r]",-1);
        Arrays.sort(lines);

        StringBuffer b = new StringBuffer();
        for(int i=0;i<lines.length;i++)
        b.append(lines[i]+"\n");
        b.deleteCharAt(b.length()-1);
        return b.toString();
    }
}
```

gus.stringtransform.line.sortlength

Transformation textuelle qui trie les lignes du texte selon leurs longueurs.

```
package gus05.entity.gus.stringtransform.line.sortlength;

import java.util.Arrays;
import java.util.Collections;
import java.util.Comparator;
import java.util.List;

import gus05.framework.core.Entity;
import gus05.framework.features.Transform;

public class TransformLineSortLength implements Entity, Transform
{
    public String getName()      {return "gus.stringtransform.line.sortlength";}
    public String getCreationDate(){return "2009.12.11";}

    public Object transform(Object obj) throws Exception
    {
        String[] lines = obj.toString().split("[\n\r]",-1);
        List list = Arrays.asList(lines);
        Collections.sort(list,comparator);

        StringBuffer b = new StringBuffer();
        for(int i=0;i<list.size();i++)
        b.append(list.get(i)+"\n");
        b.deleteCharAt(b.length()-1);
        return b.toString();
```

```
    }

    private static Comparator comparator = new Comparator(){
             public int compare(Object o1, Object o2)
             {
                    String s1 = (String) o1;
                    String s2 = (String) o2;
                    Integer n1 = new Integer(s1.length());
                    Integer n2 = new Integer(s2.length());
                    int r = n1.compareTo(n2);
                    return r!=0?r:s1.compareTo(s2);
             }};
}
```

gus.stringtransform.line.invert

Transformation textuelle qui inverse l'ordre des lignes du texte.

```
package gus05.entity.gus.stringtransform.line.invert;

import gus05.framework.core.Entity;
import gus05.framework.features.Transform;

public class TransformLineInvert implements Entity, Transform
{
    public String getName()      {return "gus.stringtransform.line.invert";}
    public String getCreationDate(){return "2006.06.21";}

    public Object transform(Object obj) throws Exception
    {
        String[] lines = obj.toString().split("[\n\r]",-1);

        StringBuffer b = new StringBuffer();
        for(int i=0;i<lines.length;i++)
        b.append(lines[lines.length-1-i]+"\n");
        b.deleteCharAt(b.length()-1);
        return b.toString();
    }
}
```

gus.stringtransform.line.shuffle

Transformation textuelle qui change aléatoirement l'ordre des lignes du texte.

```
package gus05.entity.gus.stringtransform.line.shuffle;

import java.util.Arrays;
import java.util.Collections;
import java.util.List;

import gus05.framework.core.Entity;
import gus05.framework.features.Transform;

public class TransformLineShuffle implements Entity, Transform
{
    public String getName()      {return "gus.stringtransform.line.shuffle";}
    public String getCreationDate(){return "2006.06.21";}

    public Object transform(Object obj) throws Exception
    {
```

```
        String[] lines = obj.toString().split("[\n\r]",-1);
        List list = Arrays.asList(lines);
        Collections.shuffle(list);

        StringBuffer b = new StringBuffer();
        for(int i=0;i<list.size();i++)
        b.append(list.get(i)+"\n");
        b.deleteCharAt(b.length()-1);
        return b.toString();
    }
}
```

gus.stringtransform.line.empty

Transformation textuelle qui supprime toutes les lignes vides (ou blanches) du texte.

```
package gus05.entity.gus.stringtransform.line.empty;

import gus05.framework.core.Entity;
import gus05.framework.features.Transform;

public class TransformLineEmpty implements Entity, Transform
{
    public String getName()      {return "gus.stringtransform.line.empty";}
    public String getCreationDate(){return "2006.06.21";}

    public Object transform(Object obj) throws Exception
    {
        String[] lines = obj.toString().split("[\n\r]",-1);

        StringBuffer b = new StringBuffer();
        for(int i=0;i<lines.length;i++)
        {
            StringBuffer line = new StringBuffer(lines[i]);
            while(line.length()>0 && isWhite(line.charAt(line.length()-1)))
            line.deleteCharAt(line.length()-1);

            if(line.length()>0)
            b.append(line+"\n");
        }
        b.deleteCharAt(b.length()-1);
        return b.toString();
    }

    private boolean isWhite(char c)
    {return c==' ' || c=='\t';}
}
```

gus.stringtransform.line.doublon

Transformation textuelle qui supprime toutes les lignes précédemment apparues dans le texte.

```
package gus05.entity.gus.stringtransform.line.doublon;

import java.util.HashSet;
import gus05.framework.core.Entity;
import gus05.framework.features.Transform;

public class TransformLineDoublon implements Entity, Transform
{
```

```
    public String getName()      {return "gus.stringtransform.line.doublon";}
    public String getCreationDate(){return "2006.06.21";}

    public Object transform(Object obj) throws Exception
    {
        String[] lines = obj.toString().split("[\n\r]",-1);

        StringBuffer b = new StringBuffer();
        HashSet set = new HashSet();

        for(int i=0;i<lines.length;i++)
        if(!set.contains(lines[i]))
        {
            set.add(lines[i]);
            b.append(lines[i]+"\n");
        }
        b.deleteCharAt(b.length()-1);
        return b.toString();
    }
}
```

gus.stringtransform.line.regroup

Transformation textuelle qui réduit les lignes identiques contiguës du texte à une seule ligne.

```
package gus05.entity.gus.stringtransform.line.regroup;

import gus05.framework.core.Entity;
import gus05.framework.features.Transform;

public class TransformLineRegroup implements Entity, Transform
{
    public String getName()      {return "gus.stringtransform.line.regroup";}
    public String getCreationDate(){return "2006.06.21";}

    public Object transform(Object obj) throws Exception
    {
        String[] lines = obj.toString().split("[\n\r]",-1);

        StringBuffer b = new StringBuffer();
        String line_mem = lines[0];

        for(int i=1;i<lines.length;i++)
        {
            if(!lines[i].equals(line_mem))
            {
                b.append(line_mem+"\n");
                line_mem = lines[i];
            }
        }
        b.append(line_mem);
        return b.toString();
    }
}
```

gus.stringtransform.line.index

Transformation textuelle qui fait commencer chaque ligne du texte par le numéro de ligne et une tabulation.

```java
package gus05.entity.gus.stringtransform.line.index;

import gus05.framework.core.Entity;
import gus05.framework.features.Transform;

public class TransformLineIndex implements Entity, Transform
{
    public String getName()      {return "gus.stringtransform.line.index";}
    public String getCreationDate(){return "2006.06.21";}

    public Object transform(Object obj) throws Exception
    {
        String[] lines = obj.toString().split("[\n\r]",-1);
        StringBuffer b = new StringBuffer();

        for(int i=0;i<lines.length;i++)
        b.append((i+1)+"\t"+lines[i]+"\n");
        b.deleteCharAt(b.length()-1);
        return b.toString();
    }
}
```

gus.stringtransform.line.length

Transformation textuelle qui fait commencer chaque ligne du texte par sa longueur et une tabulation.

```java
package gus05.entity.gus.stringtransform.line.length;

import gus05.framework.core.Entity;
import gus05.framework.features.Transform;

public class TransformLineLength implements Entity, Transform
{
    public String getName()      {return "gus.stringtransform.line.length";}
    public String getCreationDate(){return "2009.12.11";}

    public Object transform(Object obj) throws Exception
    {
        String[] lines = obj.toString().split("[\n\r]",-1);
        StringBuffer b = new StringBuffer();

        for(int i=0;i<lines.length;i++)
        b.append(lines[i].length()+"\t"+lines[i]+"\n");
        b.deleteCharAt(b.length()-1);
        return b.toString();
    }
}
```

gus.stringtransform.line.occ

Transformation textuelle qui fait commencer chaque ligne du texte par son nombre d'occurrences et une tabulation.

```java
package gus05.entity.gus.stringtransform.line.occ;

import java.util.HashMap;
import gus05.framework.core.Entity;
import gus05.framework.features.Transform;

public class TransformLineOcc implements Entity, Transform
```

```
{
    public String getName()      {return "gus.stringtransform.line.occ";}
    public String getCreationDate(){return "2009.12.11";}

    public Object transform(Object obj) throws Exception
    {
        String[] lines = obj.toString().split("[\n\r]",-1);

        HashMap map = new HashMap();
        for(int i=0;i<lines.length;i++)
        {
            String line = lines[i];
            if(!map.containsKey(line))
                map.put(line,new Integer(1));
            else
            {
                Integer n = (Integer)map.get(line);
                map.put(line,new Integer(n.intValue()+1));
            }
        }

        StringBuffer b = new StringBuffer();
        for(int i=0;i<lines.length;i++)
        {
             String line = lines[i];
             Integer nbOcc = (Integer)map.get(line);
             b.append(nbOcc.intValue()+"\t"+line+"\n");
        }
        if(b.length()>0) b.deleteCharAt(b.length()-1);
        return b.toString();
    }
}
```

gus.stringtransform.line.frequency

Transformation textuelle qui affiche les fréquences de chaque ligne trouvée de la plus fréquente à la moins fréquente. Chaque ligne commence par son nombre d'occurrences et une tabulation.

La fréquence d'une ligne du texte correspond au nombre d'occurrences de cette ligne dans le texte.

```
package gus05.entity.gus.stringtransform.line.frequency;

import java.util.Collections;
import java.util.Comparator;
import java.util.HashMap;
import java.util.Iterator;
import java.util.ArrayList;
import gus05.framework.core.Entity;
import gus05.framework.features.Transform;

public class TransformLineFrequency implements Entity, Transform
{
    public String getName()      {return "gus.stringtransform.line.frequency";}
    public String getCreationDate(){return "2006.06.21";}

    public Object transform(Object obj) throws Exception
    {
        String[] lines = obj.toString().split("[\n\r]",-1);

        HashMap map = new HashMap();
        for(int i=0;i<lines.length;i++)
        {
            String line = lines[i];
            if(!map.containsKey(line))
```

```
                map.put(line,new Integer(1));
            else
            {
                Integer n = (Integer)map.get(line);
                map.put(line,new Integer(n.intValue()+1));
            }
        }

        ArrayList v = new ArrayList();
        Iterator it = map.keySet().iterator();
        while(it.hasNext())
        {
            String key = (String)it.next();
            Integer val = (Integer)map.get(key);
            v.add(val.intValue()+"\t"+key);
        }

        Collections.sort(v,comparator);

        StringBuffer b = new StringBuffer();
        for(int i=v.size()-1;i>=0;i--)
        b.append(v.get(i)+"\n");
        if(b.length()>0) b.deleteCharAt(b.length()-1);
        return b.toString();
    }

    private Comparator comparator = new Comparator(){
       public int compare(Object o1, Object o2)
       {
              Integer d1 = findInt((String)o1);
              Integer d2 = findInt((String)o2);
              return d1.compareTo(d2);
       }
    };

    private Integer findInt(String s)
    {
       int n=0;
       while(Character.isDigit(s.charAt(n)))n++;
       return new Integer(s.substring(0,n));
    }
}
```

gus.stringtransform.line.repartitionfactor

Transformation textuelle qui affiche les facteurs de répartition de chaque ligne trouvée du plus grand au plus petit facteur. Chaque ligne commence par la valeur du facteur et une tabulation.

Le facteur de répartition d'une ligne du texte correspond à l'écart type des valeurs d'indice des différentes occurrences de la ligne dans le texte. Plus ce facteur est faible plus les occurrences sont regroupées dans un même endroit du texte. Plus ce facteur est élevé, plus les occurrences sont réparties uniformément dans le texte.

```
package gus05.entity.gus.stringtransform.line.repartitionfactor;

import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.HashMap;
import java.util.Iterator;
import gus05.framework.core.Entity;
import gus05.framework.features.Transform;
```

```
public class TransformLineRepartitionFactor implements Entity, Transform
{
    public String getName()      {return "gus.stringtransform.line.repartitionfactor";}
    public String getCreationDate(){return "2006.06.21";}

    public Object transform(Object obj) throws Exception
    {
        String[] lines = obj.toString().split("[\n\r]",-1);

        HashMap map = new HashMap();
        for(int i=0;i<lines.length;i++)
        {
            String line = lines[i];
            if(!map.containsKey(line))
                map.put(line,new StringBuffer(""+i));
            else
            {
                StringBuffer info = (StringBuffer)map.get(line);
                info.append(";"+i);
            }
        }

        ArrayList v = new ArrayList();
        Iterator it = map.keySet().iterator();
        while(it.hasNext())
        {
            String key = (String)it.next();
            String repartition = (String) map.get(key);
            double val = calculateFactor(repartition);
            v.add(val+"\t"+key);
        }

        Collections.sort(v,comparator);

        StringBuffer b = new StringBuffer();
        for(int i=0;i<v.size();i++)
        b.append(v.get(i)+"\n");

        if(b.length()>0)b.deleteCharAt(b.length()-1);
        return b.toString();
    }

    private double calculateFactor(String repartition)
    {
        String[] c = repartition.split(";");
        int number = c.length;
        int[] d = new int[number];

        double sum = 0;
        for(int i=0;i<number;i++)
        {
            d[i] = Integer.parseInt(c[i]);
            sum += d[i];
        }
        double moy = sum/number;

        sum = 0;
        for(int i=0;i<number;i++)
        sum += (d[i]-moy)*(d[i]-moy);

        sum = Math.sqrt(sum);
        return sum/number;
    }

    private Comparator comparator = new Comparator()
    {
      public int compare(Object o1, Object o2)
        {
            Double d1 = findDouble((String)o1);
            Double d2 = findDouble((String)o2);
            return d1.compareTo(d2);
```

```
        }
       };

    private Double findDouble(String s)
    {return new Double(s.split("\t",2)[0]);}
}
```

gus.stringtransform.line.repartitionlisting

Transformation textuelle qui affiche les répartitions de chaque ligne trouvée. Chaque ligne commence par la répartition et une tabulation.

La répartition d'une ligne du texte correspond à la succession des positions des différentes occurrences de cette ligne, séparés par des point-virgules. L'ensemble des répartitions des lignes du texte permet entre autre de reconstituer ce dernier.

```
package gus05.entity.gus.stringtransform.line.repartitionlisting;

import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.HashMap;
import java.util.Iterator;
import gus05.framework.core.Entity;
import gus05.framework.features.Transform;

public class TransformLineRepartitionListing implements Entity, Transform
{
    public String getName()      {return "gus.stringtransform.line.repartitionlisting";}
    public String getCreationDate(){return "2006.06.21";}

    public Object transform(Object obj) throws Exception
    {
        String[] lines = obj.toString().split("[\n\r]",-1);

        HashMap map = new HashMap();
        for(int i=0;i<lines.length;i++)
        {
            String line = lines[i];
            if(!map.containsKey(line))
                map.put(line,new StringBuffer(""+i));
            else
            {
                StringBuffer info = (StringBuffer)map.get(line);
                info.append(";"+i);
            }
        }

        ArrayList v = new ArrayList();
        Iterator it = map.keySet().iterator();
        while(it.hasNext())
        {
            String key = (String)it.next();
            v.add(map.get(key)+"\t"+key);
        }
        Collections.sort(v);

        StringBuffer b = new StringBuffer();
        for(int i=0;i<v.size();i++)
        b.append(v.get(i)+"\n");

        if(b.length()>0)b.deleteCharAt(b.length()-1);
        return b.toString();
    }
```

```
}
```

gus.stringtransform.line.averageposition

Transformation textuelle qui affiche les positions moyennes de chaque ligne trouvée de la plus grande à la plus petite valeur. Chaque ligne commence par la position moyenne et une tabulation.

```
package gus05.entity.gus.stringtransform.line.averageposition;

import java.util.Collections;
import java.util.Comparator;
import java.util.HashMap;
import java.util.Iterator;
import java.util.ArrayList;
import gus05.framework.core.Entity;
import gus05.framework.features.Transform;

public class TransformLineAveragePosition implements Entity, Transform
{
    public String getName()      {return "gus.stringtransform.line.averageposition";}
    public String getCreationDate(){return "2006.06.21";}

    public Object transform(Object obj) throws Exception
    {
        String[] lines = obj.toString().split("[\n\r]",-1);

        HashMap map1 = new HashMap();
        HashMap map2 = new HashMap();

        for(int i=0;i<lines.length;i++)
        {
            String line = lines[i];
            if(!map1.containsKey(line))
            {
                map1.put(line,new Integer(1));
                map2.put(line,new Integer(i));
            }
            else
            {
                Integer n1 = (Integer)map1.get(line);
                map1.put(line,new Integer(n1.intValue()+1));

                Integer n2 = (Integer)map2.get(line);
                map2.put(line,new Integer(n2.intValue()+i));
            }
        }

        ArrayList v = new ArrayList();
        Iterator it = map1.keySet().iterator();
        while(it.hasNext())
        {
            String key = (String)it.next();
            Integer number = (Integer)map1.get(key);
            Integer sum = (Integer)map2.get(key);
            double val = (double)sum.intValue() / (double)number.intValue();
            v.add(val+"\t"+key);
        }

        Collections.sort(v,comparator);

        StringBuffer b = new StringBuffer();
        for(int i=0;i<v.size();i++)
        b.append(v.get(i)+"\n");

        if(b.length()>0)b.deleteCharAt(b.length()-1);
        return b.toString();
```

```
    }

    private Comparator comparator = new Comparator()
    {
       public int compare(Object o1, Object o2)
         {
             Double d1 = findDouble((String)o1);
             Double d2 = findDouble((String)o2);
             return d1.compareTo(d2);
         }
       };

    private Double findDouble(String s)
    {return new Double(s.split("\t",2)[0]);}
}
```

gus.stringtransform.line.buildtab

```
package gus05.entity.gus.stringtransform.line.buildtab;

import java.util.ArrayList;
import gus05.framework.core.Entity;
import gus05.framework.features.Transform;

public class TransformLineBuildTab implements Entity, Transform
{
    public String getName()      {return "gus.stringtransform.line.buildtab";}
    public String getCreationDate(){return "2006.06.21";}

    public Object transform(Object obj) throws Exception
    {
        String[] lines = obj.toString().split("[\n\r]",-1);

        ArrayList v1 = new ArrayList();
        ArrayList v2 = new ArrayList();
        int dimMax = 0;

        boolean gap = false;
        for(int i=0;i<lines.length;i++)
        {
            if(lines[i].equals("")) gap = true;
            else
            {
                if(gap)
                {
                    v1.add(v2);
                    if(v2.size()>dimMax)
                        dimMax = v2.size();

                    v2 = new ArrayList();
                    gap = false;
                }
                v2.add(lines[i]);
            }
        }

        if(v2.size()>0)
        {
            v1.add(v2);
            if(v2.size()>dimMax)
                dimMax = v2.size();
        }
```

```
        int X = dimMax;
        int Y = v1.size();

        String[][] table = new String[X][Y];
        initEmptyTable(table);

        for(int j=0;j<Y;j++)
        {
             ArrayList v = (ArrayList)v1.get(j);
            for(int i=0;i<v.size();i++)
            table[i][j] = (String)v.get(i);
        }

        StringBuffer b = new StringBuffer();
        for(int i=0;i<X;i++)
        {
            b.append(table[i][0]);
            for(int j=1;j<Y;j++)
            b.append("\t"+table[i][j]);
            b.append("\n");
        }
        if(b.length()>0)b.deleteCharAt(b.length()-1);
        return b.toString();
    }

    private void initEmptyTable(String[][] table)
    {
        for(int i=0;i<table.length;i++)
        for(int j=0;j<table[0].length;j++)
            table[i][j] = "";
    }
}
```

gus.stringtransform.character.normalize

gus.stringtransform.character.normalizewhite

gus.stringtransform.calcul.addrandom

gus.stringtransform.calcul.average

gus.stringtransform.calcul.ecarttype

gus.stringtransform.calcul.formatnumber

gus.stringtransform.calcul.normalize

gus.stringtransform.calcul.sort

gus.stringtransform.calcul.sum

gus.stringtransform.cutting.character
gus.stringtransform.cutting.displaycode
gus.stringtransform.cutting.word

gus.stringtransform.encoding.decode_u1
gus.stringtransform.encoding.decode_u2
gus.stringtransform.encoding.encode_u1
gus.stringtransform.encoding.encode_u2

gus.stringtransform.generate.files
gus.stringtransform.generate.formula
gus.stringtransform.generate.lookups
gus.stringtransform.generate.multiplicateblock
gus.stringtransform.generate.multiplicateline
gus.stringtransform.generate.randoms
gus.stringtransform.generate.variables

gus.stringtransform.japanese.hiragana
gus.stringtransform.japanese.katakana

gus.stringtransform.regex.clearwebpage
gus.stringtransform.regex.javaimport
gus.stringtransform.regex.url