

# Kika:Text Preprocessor

Pablo Marchant

September 20, 2006

## Abstract

This document describes the idea of the kika software project, a text preprocessor that uses xml. All specifications mentioned here are subject to change.

## 1 Introduction

Kika is meant to simplify tedious typing on various types of text files, by specifying certain rules on an xml file, wich are performed on a text file. Up to the moment, the project is on a planning stage, but the basic idea is already decided. Two kinds of rules have been determined. One that runs over a one group of parameters (simply called, a rule), and another one that can be used to perform iterations over datasets (called an iterator)

## 2 Calling a rule

To perform a rule on certain parts of a text file, a escape string must be defined, the string that separates the parameters must be defined and the string that indicates the end of the parameters must be defined. All this definitions are written on the xml file. So, the general form of calling a rule looks like:

```
<escape String><Rule name><Parameter Separator><Param1><Parameter Separator>...<Parameter end><escape String>
```

So, for example, if I define a rule named hello that receives two parameters, specify the escape string to be #, specify parameters to be separated by a white space and the end of parameters specified by an end of line, the calling would look like:

```
#hello param1 param2  
Any text can go here...  
#
```

The text that goes between the "end of parameters string" and the "escape string" wont be processed.

## 3 Normal Rules

I call normal rules to those that receive only one set of parameters. the basic xml for these would have the form:

```

<rule name="hello">
  <rb>
    <print>My name is </print> <param id="1"/><print>\n</print>
  </rb>
  <ra>
    <print>Goodbye</print> <param id="2"/><print>\n</print>
  </ra>
</rule>

```

The rule is divided in an "ra" node, and an "rb" node. Rb stands for "right before", and Ra for "right after". What goes on <rb> will be put before the text that isn't processed, and what goes on <ra> will be put after the text that is not processed.

A print node, prints what is inside of it literally, while a param node, prints the value of the specified parameter.

So, this rule would transform the text on the call made on the "Calling a rule" section for:

```

My name is param1
Any text can go here...
Goodbye param2

```

## 4 Iteration Rules

An iteration rule is very similar to a normal one, except for the fact that it does not contain a section of unprocessed text. It's meant to iterate over datasets. For example, I could call a rule named "iter" with the text:

```

#iter param1 param2
1 2
12 32
1 43
#

```

The basic xml would be:

```

<iteration name="iter">
  <rb>
    <print>My name is </print> <param id="1"/><print>\n</print>
  </rb>
  <iterate>
    <itparam id="1"> <print> & </print> <itparam id="2" /> <print>\n</print>
  </iterate>
  <ra>
    <print>Goodbye</print> <param id="2"/><print>\n</print>
  </ra>
</iteration>

```

The "rb" and "ra" nodes have the same use than in normal rules, except that here, they enclose the result of the iteration. The "print" and "param" do exactly the same thing than before.

The "iterate" node specifies what to do in each iteration, and the "itparam" nodes refer to the parameters of each iteration.

The general parameters, those that are referred by the "param" nodes, are the ones that are given first after the name of the iteration rule. Every text that is between "end of parameters strings" is an iteration. So, the result of performing this iteration rule over the text would be:

```
My name is param1
1 & 2
12 & 32
1 & 43
Goodbye param2
```

## 5 Xml file

In the xml file, all rules and iteration rules are nested on a <rules> node. This node defines the strings that identify the rules. A complete file of xml rules would look like:

```
<rules es="#" ps=" " pe="\n">
<rule name="hello">
  <rb>
    <print>My name is </print> <param id="1"/><print>\n</print>
  </rb>
  <ra>
    <print>Goodbye</print> <param id="2"/><print>\n</print>
  </ra>
</rule>
<iteration name="iter">
  <rb>
    <print>My name is </print> <param id="1"/><print>\n</print>
  </rb>
  <iterate>
    <itparam id="1"> <print> & </print> <itparam id="2" /> <print>\n</print>
  </iterate>
  <ra>
    <print>Goodbye</print> <param id="2"/><print>\n</print>
  </ra>
</iteration>
</rules>
```

## 6 Conclusion

This is the basic idea for Kika. There are also other things that could be included, like control flow, and numeric operations, but those are still not well defined, and are not considered for a first release...