String NodeBrain Module

Release 0.8.17

String NodeBrain Module
August 2014
NodeBrain Open Source Project

Release 0.8.17

Author: Ed Trettevik

History

2012-06-04 Title: *String NodeBrain Module*
Author: Ed Trettevik <eat@nodebrain.org>
Publisher: NodeBrain Open Source Project

2012-06-04 Release 0.8.10

- This is a first edition.

Preface

This manual is intended for users of the String NodeBrain Module, a plug-in for string manipulation. The reader is expected to be familiar with the basic concepts of NodeBrain. See www.nodebrain.org for general information on NodeBrain.

Documents

NodeBrain Guide - Information on using `nb`
NodeBrain Tutorial - A gentle introduction to `nb` and the rule language
NodeBrain Language - Rule language syntax and semantics
NodeBrain Library - C API

Document Conventions

Sample code and input/output examples are displayed in a monospace font, indented in HTML and Info, and enclosed in a box in PDF or printed copies. Bold text is used to bring the reader's attention to specific portions of an example. In the following example, the first and last line are associated with the host shell and the lines in between are input or output unique to NodeBrain. The `define` command is highlighted, indicating it is the focus of the example. Lines ending with a backslash \ indicate when a command is continued on the next displayed line. This is supported by the language within source files, but not for other methods of command input. If you copy an example of a command displayed over multiple lines, you must enter it as a single line when used outside the context of a source file.

```
$ nb
> define myFirstRule on(a=1 and b=2) mood="happy";
> assert mood="sad";
> show mood
mood = "sad"
> assert a=1,b=2,c=3,d="This is an example of a long single line that",\
    e="we depict on multiple lines to fit on the documnet page";
2008/06/05 12:09:08 NB000I Rule myFirstRule fired(mood="happy")
> show mood
mood = "happy"
> quit
$
```

Table of Contents

1 Concepts

The String module implements nodes that manipulate strings in some way. This is initially a very minimal set of evaluation methods. NodeBrain is not intended to perform string manipulation with the goal of producing output for human consumption. However, it is helpful to have some string manipulation capability to consume text representations of events. Regular expressions provide the functionality needed to extract information from text. Once extracted, we sometimes need to transform the extracted strings. In general, it is best to delegate this function to servants using your favorate string manipulation language. But where we see a repeating need for a simple string manipulation, additional skills will be added to this module.

Regular expression based substitution supported by the PCRE library will most likely be included in the language in a future release. The operations provided by this module are expected to be less complex and more special purpose.

2 Tutorial

Generosity with strings is not generosity; It is a deal. —Marya Mannes

Here's the deal with NodeBrain strings. Translators and regular expressions are used to recognize and take strings apart and symbolic substitution is used to construct new strings. Tree nodes can be used to translate strings by looking up one string and substituting another. But that's about the limit of NodeBrain's generosity with strings. The string module exists to enable additional string manipulation capability as needed to digest input event attributes for processing with rules. So far, little has been needed, and this is a trivial module.

The following example illustrates how a `string.utc` node can be used to transform a time stamp in a broken down format into a UTC time string. This transformation is based on the strptime C function.

```
> define utc node string.utc;
> define whenUtc cell utc(when,"%Y-%m-%d %H:%M:%S");
> assert when="2012-06-19 10:20:35";
> assert x=whenUtc; # force evaluation of whenUtc since there is no rule depending on it
> show x;
x = "1340101235"
> assert when="2012-05-12 12:15:50";
> assert x=whenUtc;
> show x;
x = "1336824950"
```

The next example might apply in a case where a string value is to be used in an output alarm passed to an application where ";" has a special meaning. A `string.chrsub` node is used to substitute a "," for any ";" in the string.

```
> define chrsub node string.chrsub;
> define clean cell chrsub(dirty,";,");
> assert dirty="There is always hope; at least for now.";
> assert x=clean; # force evaluation of clean since there is no rule depending on it
> show x;
x = "There is always hope, at least for now."
```

3 Commands

This section describes commands used with a String node.

3.1 Define

The `define` command is used to create String manipulation nodes. Although any term can be defined for a String node, using the skill name for the term is generally a good idea.

Syntax

chrsubDefineCmd ::= **define** s̊ **chrsub** s̊ **node** s̊ **string.chrsub** [**;**] •
utcDefineCmd ::= **define** s̊ **utc** s̊ **node** s̊ **string.utc** [**;**] •

3.2 Cell Evaluation

String manipulation cells use String nodes to perform operations on string operands and/or produce strings.

Syntax

chrsubCell ::= **chrsub(** *string* **,** *pairString* **)**
utcCell ::= **utc(** *timestring* **,** *formatstring* **)**

3.2.1 chrsub

The `chrsub` node substitutes one character for another in a string.

```
strsub(text,"xy")    # character x replaces y
strsub(text,"xyab")  # character x replaces y and a replaced b
```

3.2.2 utc

The `utc` node converts a time string into UTC time. See the man page for strptime for a description of the format string.

```
utc("2001-11-12 18:31:01","%Y-%m-%d %H:%M:%S")
utc(mytimestamp,"%Y-%m-%d %H:%M:%S")
```

3.3 Assert

Assertions are not supported by this module.

3.4 Node Commands

This module does not implement node commands.

3.5 Module Commands

This module does not implement module commands.

4 Triggers

This module does not implement triggers.

Index