SWI-Prolog YAML library

Jan Wielemaker
VU University Amsterdam
CWI, Amsterdam
The Netherlands
E-mail: J.Wielemaker@vu.nl

August 26, 2018

Abstract

This package reads and writes YAML documents from and to SWI-Prolog streams, files and strings. It is based on libyaml. This C library is being used by several languages. Using this C library provides good performance, and interoperability with YALM infrastructure used by other systems.

Contents

1 yaml.pl: Process YAML data

This module parses YAML serialized data into a Prolog term with structure that is compatible with the JSON library. This library is a wrapper around the C library `libyaml`. This library forms the basis of the YAML support in several languages and thus guarantees compatibility of our YAML support with other languages.

yaml_read(+*Input, -DOM*) *[det]*

Parse *Input* to a YALM *DOM*. The *DOM* representation uses the following mapping:

- A YAML sequence is mapped to a Prolog List.
- A YAML mapping is mapped to a Prolog dict.
- Untagged *scalars* follow the implicit tag rules defined by YAML, providing numbers (int, float and special floats), `null` and the booleans `true` and `false`. Other untagged values are returned as a Prolog string. Tagged values are returned as `tag(Tag, String)` which is processed by `yalm_tagged/3`. This internal predicate calls the user hook `yaml:tagged/3` with the same arguments and, if the hook fails, provides the following defaults:
  - `!!binary` converts the Base64 to a string of bytes.
  - `!!str` explicitly keeps a string
  - `!!null` translates ”null” to `null`
  - `!!bool` translates to `true` and `false`
  - `!!int` translates to an integer
  - `!!float` translates to a float
  - Anything else is returned as `tag(Tag, String)`

Arguments

Input	is one of (1) a stream, (2) a term `string(Data)` or (3) a file name.

yaml_write(+*Out:stream, +DOM*) *[det]*
yaml_write(+*Out:stream, +DOM, +Options*) *[det]*

Emit a YAML *DOM* object as a serialized YAML document to the stream *Out*. *Options* processed are:

canonical(+*Boolean*)
Use canonical representation. Default is `false`.

unicode(+*Boolean*)
Use unicode Default is `true`.

implicit(+*Boolean*)
Use implicit or explicit representation. Currently only affects the opening and closing the document. Default is `true`. Use `false` for embedded documents.

factorize(+*Boolean*)
If `true`, minimize the term by factoring out common structures and use `&anchor` and `*anchor`. Factorization is always used if *DOM* is a cyclic term.

tagged(+*Tag, ?String, ?Value*) *[semidet,multifile]*

Hook that allows convering `!!tag` values to be decoded or encoded.

Index