brianmario / yajl-ruby Public

Code Issues 52 Pull requests 19 Actions Projects Security 1 ...

7168bd79b8

yajl-ruby / ext / yajl / yajl_buf.c

brianmario move to rake-compiler, Bundler History

1 contributor

119 lines (102 sloc) | 3.21 KB

```c
/*
 * Copyright 2010, Lloyd Hilaiel.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are
 * met:
 *
 *  1. Redistributions of source code must retain the above copyright
 *     notice, this list of conditions and the following disclaimer.
 *
 *  2. Redistributions in binary form must reproduce the above copyright
 *     notice, this list of conditions and the following disclaimer in
 *     the documentation and/or other materials provided with the
 *     distribution.
 *
 *  3. Neither the name of Lloyd Hilaiel nor the names of its
 *     contributors may be used to endorse or promote products derived
 *     from this software without specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS'' AND ANY EXPRESS OR
 * IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
 * WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
 * DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT,
 * INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
 * (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
 * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGE.
 */

#include "yajl_buf.h"

#include <assert.h>
#include <stdlib.h>
#include <string.h>

#define YAJL_BUF_INIT_SIZE 2048

struct yajl_buf_t {
    unsigned int len;
    unsigned int used;
    unsigned char * data;
    yajl_alloc_funcs * alloc;
};

static
void yajl_buf_ensure_available(yajl_buf buf, unsigned int want)
{
    unsigned int need;

    assert(buf != NULL);

    /* first call */
    if (buf->data == NULL) {
        buf->len = YAJL_BUF_INIT_SIZE;
        buf->data = (unsigned char *) YA_MALLOC(buf->alloc, buf->len);
        buf->data[0] = 0;
    }

    need = buf->len;

    while (want >= (need - buf->used)) need <<= 1;

    if (need != buf->len) {
        buf->data = (unsigned char *) YA_REALLOC(buf->alloc, buf->data, need);
        buf->len = need;
    }
}

yajl_buf yajl_buf_alloc(yajl_alloc_funcs * alloc)
{
    yajl_buf b = YA_MALLOC(alloc, sizeof(struct yajl_buf_t));
    memset((void *) b, 0, sizeof(struct yajl_buf_t));
    b->alloc = alloc;
    return b;
}
```

```c

void yajl_buf_free(yajl_buf buf)
{
    assert(buf != NULL);
    if (buf->data) YA_FREE(buf->alloc, buf->data);
    YA_FREE(buf->alloc, buf);
}

void yajl_buf_append(yajl_buf buf, const void * data, unsigned int len)
{
    yajl_buf_ensure_available(buf, len);
    if (len > 0) {
        assert(data != NULL);
        memcpy(buf->data + buf->used, data, len);
        buf->used += len;
        buf->data[buf->used] = 0;
    }
}

void yajl_buf_clear(yajl_buf buf)
{
    buf->used = 0;
    if (buf->data) buf->data[buf->used] = 0;
}

const unsigned char * yajl_buf_data(yajl_buf buf)
{
    return buf->data;
}

unsigned int yajl_buf_len(yajl_buf buf)
{
    return buf->used;
}

void
yajl_buf_truncate(yajl_buf buf, unsigned int len)
{
    assert(len <= buf->used);
    buf->used = len;
}
```