

The logo for RT-Thread, featuring a blue stylized wave icon above the text "RT-Thread" in a bold, blue, sans-serif font. The logo is centered within a white, rounded rectangular shape that has a slight 3D effect and is surrounded by several orange and white circles of varying sizes on a blue background.

RT-Thread

网络编程提升篇

HTTP：利用 HTTP 协议获取天气

目录

- 背景介绍
- 运行示例
- 注意事项
- 示例代码讲解



背景介绍

背景介绍

- HTTP 协议是互联网上应用最为广泛的一种网络协议，越来越多的应用程序需要直接通过 HTTP 协议来访问网络资源。
- webclient 是 RT-Thread 上实现的一个 HTTP 客户端，用来提供高效且功能丰富的 HTTP 客户端编程工具包。
- 这个教程展示了如何利用 HTTP 协议获取天气，我们是使用 webclient 这个工具包实现的。



运行示例

开启 WebClient 软件包

- [Webclient](#) 软件包是一个 HTTP 协议的客户端工具，利用这个工具可以完成与 HTTP 服务器的通信。
- 打开 env 工具输入 `menuconfig` 按照下面的路径开启 WebClient 软件包

```
RT-Thread online packages
IoT - internet of things --->
[*] WebClient: A webclient package for rt-thread --->
```

- WebClient 软件包的配置如下所示

```
--- WebClient: A HTTP/HTTPS Client for RT-Thread
[ ] Enable support tls protocol
[ ] Enable webclient GET/POST samples
Version (latest) --->
```

WebClient 配置介绍

- Eable support tls protocol: 开启加密传输
- Enable webclient GET/POST samples: 开启 webclient 的 GET/POST 示例程序
- version (latest) --->: 选择版本

开启 cJSON 软件包

- 因为服务器返回的天气信息是 JSON 格式的，所以我们需要利用 cJSON 软件包来解析返回的数据。
- 打开 env 工具输入 `menuconfig` 按照下面的路径开启 [cJSON](#) 软件包

```
RT-Thread online packages
IoT - internet of things --->
[*] cJSON: Ultralightweight JSON parser in ANSI C --->
```

- cJSON 软件包的配置保持默认即可

```
--- cJSON: Ultralightweight JSON parser in ANSI C
    Version (v1.0.2) --->
```


获取示例代码

- 打开 `env` 工具输入 `menuconfig` 按照下面的路径获取 [httpclient.c](#) 示例代码

```
RT-Thread online packages --->
  miscellaneous packages --->
    samples: RT-Thread kernel and components samples --->
      a network_samples package for rt-thread --->
        [*] [network] http client
```

- 保存并更新软件包 `pkgs --update`
- 编译工程 `scons`
- 然后运行 `qemu`

运行示例代码

- 示例代码中已经将 `weather` 命令导出到了 `msh` 命令列表中，因此系统运行起来后，在 `msh` 命令行下输入 `weather` 命令即可让示例代码运行。

```
msh> weather
```

- 终端会打印出一些天气信息

```
cityName:浦东  
temp :25°C  
wd :东风  
ws :2级  
sd :49%  
date :20131012  
time :15:00
```

- **注:**由于 `env` 编码格式不支持 `UTF-8`，会出现中文显示乱码的问题。



注意事项

注意事项

- 示例代码中的 `AREA_ID` 是城市的代码，更换城市代码可以获取不同城市的天气。
- 由于 `env` 编码格式不支持 `UTF-8`，会出现中文显示乱码的问题。
- 此程序仅为 `http client` 的示例程序，其中获取天气的 `API` 已经被弃用，实际使用时更换为其他的 `API` 才可获取最新的天气。
- 电脑需要关闭防火墙



示例代码讲解

示例代码讲解

```
/*
 * 程序清单：利用 http client 获取天气
 *
 * 这是一个利用 http client 获取天气的例程
 * 导出 weather 命令到控制终端
 * 命令调用格式：weather
 * 无参数
 * 程序功能：作为一个 http 客户端，通过 GET 方法与远方服务器通信，获取到服务器传来的天气信息
 */

#include <webclient.h> /* 使用 HTTP 协议与服务器通信需要包含此头文件 */
#include <sys/socket.h> /* 使用BSD socket， 需要包含socket.h头文件 */
#include <netdb.h>
#include <cJSON.h>
#include <finsh.h>

#define GET_HEADER_BUFSZ 1024 //头部大小
#define GET_RESP_BUFSZ 1024 //响应缓冲区大小
#define GET_URL_LEN_MAX 256 //网址最大长度
#define GET_URI "http://mobile.weather.com.cn/data/sk/%s.html" //获取天气的 API
#define AREA_ID "101021300" //上海浦东地区 ID
```

示例代码讲解

```
/* 天气数据解析 */
void weather_data_parse(rt_uint8_t *data)
{
    cJSON *root = RT_NULL, *object = RT_NULL, *item = RT_NULL;

    root = cJSON_Parse((const char *)data);
    if (!root)
    {
        rt_kprintf("No memory for cJSON root!\n");
        return;
    }
    object = cJSON_GetObjectItem(root, "sk_info");

    item = cJSON_GetObjectItem(object, "cityName");
    rt_kprintf("\ncityName:%s ", item->valuelstring);

    item = cJSON_GetObjectItem(object, "temp");
    rt_kprintf("\ntemp   :%s ", item->valuelstring);

    item = cJSON_GetObjectItem(object, "wd");
    rt_kprintf("\nwd     :%s ", item->valuelstring);
}
```

示例代码讲解

```
item = cJSON_GetObjectItem(object, "ws");
rt_kprintf("\nws   :%s ", item->valuestring);

item = cJSON_GetObjectItem(object, "sd");
rt_kprintf("\nsd   :%s ", item->valuestring);

item = cJSON_GetObjectItem(object, "date");
rt_kprintf("\ndate  :%s", item->valuestring);

item = cJSON_GetObjectItem(object, "time");
rt_kprintf("\ntime  :%s \n", item->valuestring);

if (root != RT_NULL)
    cJSON_Delete(root);
}
void weather(int argc, char **argv)
{
    rt_uint8_t *buffer = RT_NULL;
    int resp_status;
    struct webclient_session *session = RT_NULL;
    char *weather_url = RT_NULL;
    int content_length = -1, bytes_read = 0;
    int content_pos = 0;
```


示例代码讲解

```
/* 为 weather_url 分配空间 */
weather_url = rt_calloc(1, GET_URL_LEN_MAX);
if (weather_url == RT_NULL)
{
    rt_kprintf("No memory for weather_url!\n");
    goto __exit;
}
/* 拼接 GET 网址 */
rt_snprintf(weather_url, GET_URL_LEN_MAX, GET_URI, AREA_ID);

/* 创建会话并且设置头部的大小 */
session = webclient_session_create(GET_HEADER_BUFSZ);
if (session == RT_NULL)
{
    rt_kprintf("No memory for get header!\n");
    goto __exit;
}

/* 发送 GET 请求使用默认头部 */
if ((resp_status = webclient_get(session, weather_url)) != 200)
{
    rt_kprintf("webclient GET request failed, response(%d) error.\n", resp_status);
    goto __exit;
}
```

示例代码讲解

```
/* 分配用于存放接收数据的缓冲 */
buffer = rt_calloc(1, GET_RESP_BUFSZ);
if(buffer == RT_NULL)
{
    rt_kprintf("No memory for data receive buffer!\n");
    goto __exit;
}

content_length = webclient_content_length_get(session);
if (content_length < 0)
{
    /* 返回的数据是分块传输的. */
    do
    {
        bytes_read = webclient_read(session, buffer, GET_RESP_BUFSZ);
        if (bytes_read <= 0)
        {
            break;
        }
    } while (1);
} else{
    do
    {
```

示例代码讲解

```
        bytes_read = webclient_read(session, buffer,
        content_length - content_pos > GET_RESP_BUFSZ ?
        GET_RESP_BUFSZ : content_length - content_pos);
    if (bytes_read <= 0)
    {
        break;
    }
    content_pos += bytes_read;
} while (content_pos < content_length);
}
/* 天气数据解析 */
weather_data_parse(buffer);
exit:
/* 释放网址空间 */
if (weather_url != RT_NULL)
    rt_free(weather_url);
/* 关闭会话 */
if (session != RT_NULL)
    webclient_close(session);
/* 释放缓冲区空间 */
if (buffer != RT_NULL)
    rt_free(buffer);
}
MSH_CMD_EXPORT(weather, Get weather by webclient);
```