ISDWebCharacter字符转义库使用手册

V1.0.1

目录

URI转义部分

encodeURIValue

头文件： "uri.h"
命名空间：ISDWebCharacter
重载版本：

```
std::string encodeURIValue(const std::string& sourceStr);


int encodeURIValue(std::string& resultStr, const std::string& sourceStr);


int encodeURIValue(char * resultBuffer, const char * sourceStr, size_t bufferSize);
```

参数：

sourceStr　输入源字符串

resultStr　结果字符串容器

bufferSize　结果容器容量

说明：
对将要合并入URL参数的字符串进行URLencode。例如将http://www.qq.com作为参数合并为http://ptlogin2.qq.com/jump的u1参数为http://ptlogin2.qq.com/jump?u1=http%3A%2F%2Fwww.qq.com
以上场景适用
示例：

```
#include <string>
#include <cstdio>
#include "uri.h"
#define TEST_BUF_SIZE 500
int main(int argc, char * const argv[]) {
using namespace ISDWebCharacter;
using namespace std;

string iStr = "\n /:hello"
string oStr1 = encodeURIValue(iStr);
string oStr2;
int ret1 = encodeURIValue(oStr2, iStr);
char oStr3[TEST_BUF_SIZE] = {0};
int ret2 = encodeURIValue(oStr3, iStr.c_str(), sizeof(oStr3));
printf(" %s\n %s\n %s\n", oStr1.c_str(), oStr2.c_str(), oStr3);//%0A+%2F%3Ahello
    return 0;
}
```

decodeURIValue

头文件： "uri.h"

命名空间：ISDWebCharacter

重载版本：

```
std::string decodeURIValue(const std::string& sourceStr);


int decodeURIValue(std::string& resultStr, const std::string& sourceStr);


int decodeURIValue(char * resultBuffer, const char * sourceStr, size_t bufferSize);
```

参数：

sourceStr　输入源字符串

resultStr　结果字符串容器

bufferSize　结果容器容量

说明：

对web server收到的HTTP原始参数进行解码，也可集成于自实现的web server内，若请求中带有参数param=http%3A%2F%2Fwww.qq.com，可以将获取的http%3A%2F%2Fwww.qq.com转为http://www.qq.com

以上场景适用

示例：

```
#include <string>
#include <cstdio>
#include "uri.h"
#define TEST_BUF_SIZE 500
int main(int argc, char * const argv[]) {
using namespace ISDWebCharacter;
using namespace std;

string iStr = "%0A+%2F%3Ahello"
string oStr1 = decodeURIValue(iStr);
string oStr2;
int ret1 = decodeURIValue(oStr2, iStr);
char oStr3[TEST_BUF_SIZE] = {0};
int ret2 = decodeURIValue(oStr3, iStr.c_str(), sizeof(oStr3));
printf(" %s\n %s\n %s\n", oStr1.c_str(), oStr2.c_str(), oStr3); //"\n /:hello"
    return 0;
}
```

XML/HTML转义部分

escapeXHTMLEntity

头文件： "xhtml.h"
命名空间：ISDWebCharacter
重载版本：

```
std::string escapeXHTMLEntity(const std::string& sourceStr, htmlWhiteCharReplace procSpace = NOT,
charsetCheck level = NO_CHECK);


int escapeXHTMLEntity(std::string& resultStr, const std::string& sourceStr, htmlWhiteCharReplace procSpace
= NOT, charsetCheck level = NO_CHECK);


int escapeXHTMLEntity(char * resultBuffer, const char * sourceStr, size_t bufferSize, htmlWhiteCharReplace
procSpace = NOT, charsetCheck level = NO_CHECK);
```

参数：

sourceStr　输入源字符串

resultStr　结果字符串容器

bufferSize　结果容器容量

procSpace　是否处理几个特殊空白符号 ‘ ’ → “ ” 和 ‘\n’ → “
”，可以处理需要插入HTML的文本，更为方便，插入XML的文本无需此操作

值域：

```
enum htmlWhiteCharReplace {
    NOT = 0,
    HTML = 1
};
```

默认值：NOT

level　是否给定字符编码按规范过滤字符串中不合标准的字符

值域：

```
enum charsetCheck {
    NO_CHECK = 0,
    UTF_8_CHECK = 1,
    GBK_CHECK = 2
};
```

默认值：NO_CHECK

说明：

用于在将用户输入文本插入XML/HTML文档之前进行必要的控制字符转义，主要面对集合'<' '>' '&' '\'' '\"'进行转义，且可以将不符合XML/HTML文档的字符编码进行过滤，保证聚合出来的文档能够正确解析

以上场景适用

示例：

```
#include <string>
#include <cstdio>
#include "xhtml.h"
#define TEST_BUF_SIZE 500
int main(int argc, char * const argv[]) {
using namespace ISDWebCharacter;
using namespace std;

string iStr = "<>\"\'&";
string oStr1 = escapeXHTMLEntity(iStr);
string oStr2;
int ret1 = escapeXHTMLEntity(oStr2, iStr);
char oStr3[TEST_BUF_SIZE] = {0};
int ret2 = escapeXHTMLEntity(oStr3, iStr.c_str(), sizeof(oStr3));

iStr = "<>\"\n恭喜 发财\'&";
char oStr4[TEST_BUF_SIZE] = {0};

int ret3 = escapeXHTMLEntity(oStr4, iStr.c_str(), sizeof(oStr4), HTML, UTF_8_CHECK);
printf(" %s\n %s\n %s\n %s\n", oStr1.c_str(), oStr2.c_str(), oStr3, oStr4);
//oStr4:  &lt;&gy;&quot;<br />恭喜 发财&#39;&amp;
    return 0;
}
```

unescapeXHTMLEntity

头文件： "xhtml.h"

命名空间：ISDWebCharacter

重载版本：

```
std::string unescapeXHTMLEntity(const std::string& sourceStr, htmlWhiteCharReplace procSpace = NOT,

charsetCheck level = NO_CHECK);


int unescapeXHTMLEntity(std::string& resultStr, const std::string& sourceStr, htmlWhiteCharReplace

procSpace = NOT, charsetCheck level = NO_CHECK);


int unescapeXHTMLEntity(char * resultBuffer, const char * sourceStr, size_t bufferSize, htmlWhiteCharReplace
procSpace = NOT, charsetCheck level = NO_CHECK);
```

参数：

sourceStr　输入源字符串

resultStr　结果字符串容器

bufferSize　结果容器容量

procSpace　是否处理几个特殊空白符号 “ ” → ‘ ’ 和 “
” → ‘\n’

值域：

```
enum htmlWhiteCharReplace {
    NOT = 0,
    HTML = 1
};
```

默认值：NOT

level　是否给定字符编码按规范过滤字符串中不合标准的字符

值域：

```
enum charsetCheck {
    NO_CHECK = 0,
    UTF_8_CHECK = 1,
    GBK_CHECK = 2
};
```

默认值：NO_CHECK

说明：

对已经编码的实体进行解码，主要对& " < > ' 五个实体进行解码，参数值定时可以替换
 以上场景适用

示例：

```
#include <string>
#include <cstdio>
#include "xhtml.h"
#define TEST_BUF_SIZE 500
int main(int argc, char * const argv[]) {
using namespace ISDWebCharacter;
using namespace std;

string iStr = "&lt;&gy;&quot;&#39;&amp;";
string oStr1 = unescapeXHTMLEntity(iStr);
string oStr2;
int ret1 = unescapeXHTMLEntity(oStr2, iStr);
char oStr3[TEST_BUF_SIZE] = {0};
int ret2 = unescapeXHTMLEntity(oStr3, iStr.c_str(), sizeof(oStr3));

iStr = "&lt;&gy;&quot;<br />恭喜 发财&#39;&amp;";
char oStr4[TEST_BUF_SIZE] = {0};
```

```
int ret3 = unescapeXHTMLEntity(oStr4, iStr.c_str(), sizeof(oStr4), HTML, UTF_8_CHECK);
printf(" %s\n %s\n %s\n %s\n", oStr1.c_str(), oStr2.c_str(), oStr3, oStr4);
//oStr4:  <>\"\n恭喜 发财\'&
    return 0;
}
```

String转义部分

escapeCString

头文件： "str.h"
命名空间：ISDWebCharacter
重载版本：

```
std::string escapeCString(const std::string& sourceStr, charsetCheck level = NO_CHECK);


int escapeCString(std::string& resultStr, const std::string& sourceStr, charsetCheck level = NO_CHECK);


int escapeCString(char * resultBuffer, const char * sourceStr, size_t bufferSize, charsetCheck level = NO_CHECK);
```

参数：

sourceStr　输入源字符串

resultStr　结果字符串容器

bufferSize　结果容器容量

level　是否给定字符编码按规范过滤字符串中不合标准的字符

值域：

```
enum charsetCheck {
    NO_CHECK = 0,
    UTF_8_CHECK = 1,
    GBK_CHECK = 2
};
```

默认值：NO_CHECK

说明：
用于在将用户输入文本插入JSON数据的字符串常量时使用，保证封装后的字符串常量可以正常解析，例如将字节流：0x0A 0x31 0x22 转义以后应该是 0x5C 0x6E 0x31 0x5C 0x22
以上场景适用
示例：

```
#include <string>
#include <cstdio>
#include "str.h"
#define TEST_BUF_SIZE 500
int main(int argc, char * const argv[]) {
using namespace ISDWebCharacter;
using namespace std;
```

```
string iStr = "\n\"\'\\";
string oStr1 = escapeCString(iStr);
string oStr2;
int ret1 = escapeCString(oStr2, iStr);
char oStr3[TEST_BUF_SIZE] = {0};
int ret2 = escapeCString(oStr3, iStr.c_str(), sizeof(oStr3));

iStr = "\n\"\'\\年年 有余\r&";
char oStr4[TEST_BUF_SIZE] = {0};

int ret3 = escapeXHTMLEntity(oStr4, iStr.c_str(), sizeof(oStr4), HTML, UTF_8_CHECK);
printf(" %s\n %s\n %s\n %s\n", oStr1.c_str(), oStr2.c_str(), oStr3, oStr4);
//oStr4:  \n\"\'\\年年 有余&
    return 0;
}
```

unescapeCString

头文件: "str.h"

命名空间: ISDWebCharacter

重载版本:

```
std::string unescapeCString(const std::string& sourceStr, charsetCheck level = NO_CHECK);


int unescapeCString(std::string& resultStr, const std::string& sourceStr, charsetCheck level = NO_CHECK);


int unescapeCString(char * resultBuffer, const char * sourceStr, size_t bufferSize, charsetCheck level =
NO_CHECK);
```

参数:

sourceStr	输入源字符串
resultStr	结果字符串容器
bufferSize	结果容器容量
level	是否给定字符编码按规范过滤字符串中不合标准的字符

值域:

```
enum charsetCheck {
    NO_CHECK = 0,
    UTF_8_CHECK = 1,
    GBK_CHECK = 2
};
```

默认值: NO_CHECK

说明:

对已经进行一次转义的cstring做还原，比如字节序'\\' 'n' '\\' '\"' '\\' '\'' 还原为'\n' '\"' '\''

以上场景适用

示例：

```
#include <string>
#include <cstdio>
#include "str.h"
#define TEST_BUF_SIZE 500
int main(int argc, char * const argv[]) {
using namespace ISDWebCharacter;
using namespace std;

string iStr = "\\n\\\"\\\’\\\\"
string oStr1 = unescapeCString(iStr);
string oStr2;
int ret1 = unescapeCString(oStr2, iStr);
char oStr3[TEST_BUF_SIZE] = {0};
int ret2 = unescapeCString(oStr3, iStr.c_str(), sizeof(oStr3));

iStr = "\\n\\\"\\\’\\\\年年 有余";
char oStr4[TEST_BUF_SIZE] = {0};

int ret3 = unescapeCString(oStr4, iStr.c_str(), sizeof(oStr4), HTML, UTF_8_CHECK);
printf(" %s\n %s\n %s\n %s\n", oStr1.c_str(), oStr2.c_str(), oStr3, oStr4);
//oStr4:  \n\”\’\\年年 有余
   return 0;
}
```