Tencent

互联网统一公共库项目

字符转义部分说明

策划人：scorpionxu

2009年7月2日

Tencent

目录

Tencent

概述

前言

BU后台字符转义类公共库长期以来存在一些问题，对于字符的各种转义和编码操作都不符合标准。长期以来各个业务团队为使用也是煞费苦心，生产了不同的各种版本。

目前互联网BU已经是国内互联网界有着最广大用户的服务团队，在如此海量的服务挑战面前，更需要我们的技术底层有合理的基础，为了减少字符转义操作可能产生的问题，使其更加易于维护，现在我们需要统一标准

范围

本部分旨在统一

- URI编解码
- XML/HTML entity转义和还原
- String (C string / JS string)转义和还原

的标准接口，避免出现转义范围不统一，转义层数不一致导致的问题

引用标准

URI部分以RFC1738为标准

XML/HTML部分以W3C entity replace为标准

String escape以ANSI C作为依据

关键字

URI部分以RFC1738为标准

XML/HTML部分以W3C entity replace为标准

String escape以ANSI C作为依据

Tencent

接口应用场景

HTTP协议中的URL/参数/数据提交构造

如果你需要使用CGI进行一个POST请求的构造和发送，那么你必须面对所提交数据的编码，你当然不能直接在里面写：(换行) / 等等HTTP协议中本身的控制字符，而且你也不能直接就把中文等ASCII以外的字节流不经编码就放进去，这样你的请求有可能会在传输过程中的任何一个路由上被拒绝

你是否经常面对需要构造HTTP 302响应，对请求进行跳转操作？当一个URL中的某一个参数值也是一个URL的时候，你是否有做编码？

以上都是重要是使用场景，主要针对将各种数据放入HTTP req/rsp时的编解码处理

XML数据构造

将用户数据组织在XML结构中，由于用户的数据千变万化，如何保证XML能够被正确解析，需要将用户数据中的控制字符进行转义

HTML文档生成

直接生成用户需要的页面时，常常需要在页面中插入用户数据，昵称、用户输入都是常见的内容，如何保证显示正确，不被注入，需要插入HTML结构前进行转义

String转义

当要将用户元数据封装成JSON中的Javascript字符串常量时，需要使用string转义。在字符串常量中当然不能有换行、制表、引号等字符的出现

接口定义

接口包命名空间

```
namespace ISDWebCharacter{};
```

常量

用以调用接口时声明数据的Byte编码方式

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

注：只支持4 Bytes级别的标准UTF-8编码规范（见 RFC 3629）

用以声明是否在HTML转义中处理连续空格和换行符号

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

所有转义函数的返回类型定义

```
enum errorCode {
    ERR_NULL_PARAMS = -100,
    ERR_BUFFER_TOO_SMALL = -99,
    ERR_ENCODE = -98,
```

```
        OK = 0
    };
```

用于加速的字符映射表

```
    static encodeURIDefineNode URIEncodeMap[256];
    static decodeURIDefineNode URIDecodeMap[103];
    static escapeXHTMLDefineNode XHTMLEscapeMap[128];
    static escapeXHTMLDefineNode XHTMLEscapeMapNoWhite[128];
    static escapeCStringDefineNode CStringEscapeMap[128];
```

URI编解码接口

策略

只针对将要合入URL作为参数值传递的数据进行处理

ASCII范围内(0x7F以内单字节)

0x01 - 0x1F

全部进行编码

变为 '%' + ASCII HEX Digit形式: 如 0x0A (换行符) 转义为 %0A(Hex Digit统一大写)

0x20 - 0x7F

编码规范如下 (只列出需要编码的字符)

(space) ---- +

" ---- %22

```
#               ---- %23
$               ---- %24
%               ---- %25
&               ---- %26
+               ---- %2B
,               ---- %2C
/               ---- %2F
:               ---- %3A
;               ---- %3B
<               ---- %3C
=               ---- %3D
>               ---- %3E
?               ---- %3F
@               ---- %40
[               ---- %5B
\               ---- %5C
]               ---- %5D
^               ---- %5E
`               ---- %60
{               ---- %7B
|               ---- %7C
}               ---- %7D
```

0x80 – 0xFF

全部进行编码，方法与0x01 – 0x1F的方式相同

解码策略

这里需要注意一点，解码时会解码所有的 % 转义序列，如 “%36” 也会转回为 ‘6’

因此，URI编解码接口是不对称的一族接口

编码接口族

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

参数定义：

sourceStr　　传入的原串，常为HTTP参数值

返回：

结果字符串

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

参数定义：

resultStr　　结果容器字符串

sourceStr　　传入的原串，常为HTTP参数值

返回：

结果码，见enum errorCode类型定义

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

参数定义：

resultStr　　结果容器字符串缓冲区

sourceStr　　传入的原串，常为HTTP参数值

bufferSize　　结果缓冲区的大小，这里必须准确给出，否则有危险

返回：

结果码，见enum errorCode类型定义

解码接口族

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

参数定义：

sourceStr　　传入的原串，常为编码后的HTTP参数值

返回：

结果字符串

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

参数定义：

resultStr　　结果容器字符串

sourceStr　　传入的原串，　常为编码后的HTTP参数值

返回：

结果码，见enum errorCode类型定义

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

参数定义：

resultStr　　结果容器字符串缓冲区

sourceStr　　传入的原串，　常为编码后的HTTP参数值

bufferSize　　结果缓冲区的大小，这里必须准确给出，否则有危险

返回：

结果码，见enum errorCode类型定义

XML/HTML entity转义还原接口

策略

原则上一定要对以下字符进行转义

```
&        &amp;        (应该最先转毕)
<        &lt;
>        &gt;
"        &quot;
'        &#39;        (本应该是 &apos; 无奈IE不解析...)
```

在插入HTML文档时如果需要还原空格和换行效果(放在<pre> tag中时实体换行和空格能被如实表现)

```
(space)          (其实 解释出来已经不是原来的换行符了，这里要注意)
(tab)     
(LF)     <br />       (需要在转换'<' '>'字符后进行)
```

其他控制字符全部丢弃

转义接口族

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

参数定义:

sourceStr　传入的原串

procSpace　是否处理字符串中的空白字符，如(space) (LF)，默认为不处理

charsetCheck　调用者自我指明字符串的编码类型，为默认值NO_CHECK时，不处理，若指定字符集，则进行检查纠错，不符合编码规范的Byte序列将被替换为等长的'?'字符

返回:

结果字符串

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

参数定义：

resultStr　结果容器字符串

sourceStr　传入的原串

procSpace　是否处理字符串中的空白字符，如(space) (LF)，默认为不处理

charsetCheck 调用者自我指明字符串的编码类型，为默认值NO_CHECK时，不处理，若指定字符集，则进行检查纠错，不符合编码规范的Byte序列将被替换为等长的'?'字符

返回：

结果码，见enum errorCode类型定义

---

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

参数定义：

resultStr　结果容器字符串

sourceStr　传入的原串

bufferSize　结果缓冲区的大小，这里必须准确给出，否则有危险

procSpace　是否处理字符串中的空白字符，如(space) (LF)，默认为不处理

charsetCheck 调用者自我指明字符串的编码类型，为默认值NO_CHECK时，不处理，若指定字符集，则进行检查纠错，不符合编码规范的Byte序列将被替换为等长的'?'字符

返回：

结果码，见enum errorCode类型定义

还原接口族

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

参数定义：

sourceStr　传入的原串

procSpace　是否处理字符串中的空白字符，如(space) (LF)，默认为不处理

charsetCheck 调用者自我指明字符串的编码类型，为默认值NO_CHECK时，不处理，若指定字符集，则进行检查纠错，不符合编码规范的Byte序列将被替换为等长的'?'字符

返回：

结果字符串

---

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

参数定义：

resultStr 结果容器字符串

sourceStr 传入的原串

procSpace 是否处理字符串中的空白字符，如(space) (LF)，默认为不处理

charsetCheck 调用者自我指明字符串的编码类型，为默认值NO_CHECK时，不处理，若指定字符集，则进行检查纠错，不符合编码规范的Byte序列将被替换为等长的'?'字符

返回：

结果码，见enum errorCode类型定义

---

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

参数定义：

resultStr 结果容器字符串

sourceStr 传入的原串

bufferSize 结果缓冲区的大小，这里必须准确给出，否则有危险

procSpace 是否处理字符串中的空白字符，如(space) (LF)，默认为不处理

charsetCheck 调用者自我指明字符串的编码类型，为默认值NO_CHECK时，不处理，若指定字符集，则进行检查纠错，不符合编码规范的Byte序列将被替换为等长的'?'字符

返回：

结果码，见enum errorCode类型定义

String转义还原接口

策略

原则上一定要对以下字符进行转义

\	\\
"	\"
'	\'
(LF)	\n
(tab)	\t

其他控制字符全部丢弃

转义接口族

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

参数定义：

sourceStr 传入的原串

charsetCheck 调用者自我指明字符串的编码类型，为默认值NO_CHECK时，不处理，若指定字符集，则进行检查纠错，不符合编码规范的Byte序列将被替换为等长的'?'字符

返回：

结果字符串

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

参数定义：

resultStr 结果容器字符串

sourceStr 传入的原串

charsetCheck 调用者自我指明字符串的编码类型，为默认值NO_CHECK时，不处理，若指定字符集，则进行检查纠错，不符合编码规范的Byte序列将被替换为等长的'?'字符

返回：

结果码，见enum errorCode类型定义

---

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

参数定义：

resultStr 结果容器字符串

sourceStr 传入的原串

bufferSize 结果缓冲区的大小，这里必须准确给出，否则有危险

charsetCheck 调用者自我指明字符串的编码类型，为默认值NO_CHECK时，不处理，若指定字符集，则进行检查纠错，不符合编码规范的Byte序列将被替换为等长的'?'字符

返回：

结果码，见enum errorCode类型定义

还原接口族

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

参数定义：

sourceStr 传入的原串

charsetCheck 调用者自我指明字符串的编码类型，为默认值NO_CHECK时，不处理，若指定字符集，则进行检查纠错，不符合编码规范的Byte序列将被替换为等长的'?'字符

返回：

结果字符串

---

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

参数定义：

resultStr 结果容器字符串

sourceStr 传入的原串

charsetCheck 调用者自我指明字符串的编码类型，为默认值NO_CHECK时，不处理，若指定字符集，则进行检查纠错，不符合编码规范的Byte序列将被替换为等长的'?'字符

返回：

结果码，见enum errorCode类型定义

---

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

参数定义：

resultStr 结果容器字符串

sourceStr 传入的原串

bufferSize 结果缓冲区的大小，这里必须准确给出，否则有危险

charsetCheck 调用者自我指明字符串的编码类型，为默认值NO_CHECK时，不处理，若指定字符集，则进行检查纠错，不符合编码规范的Byte序列将被替换为等长的'?'字符

返回：

结果码，见enum errorCode类型定义
