转载 真香号 于 2022-02-11 16:25:28 发布 3664 收藏 2 版权
分类专栏： 网络编程 文章标签： http https java 乱码

http或https 请求的参数，为什么要urlEncode编码呢

http协议中参数的传递是"key=value"这种键值对的形式拼接，如果要传多个参数就需要用"&" 符号对键值对进行分割，如："?name1=value1&name2=value2",这样在服务端收到这种字符串时，会用"&"分割出每一个参数，然后再用"=" 分割出每一个参数的值，在计算机中用ASCII码表示。

如果我的参数值中就包含=或& 这种特殊字符串怎么办呢?

比如"name1=value1"，其中value1的值是"va&lu=e1"字符串，那么在实际传输过程中就会变成"name1=va&lu=e1"。我们本意是就只有一个键值对，但是浏览器会解析成两个键值对，这样就产生了歧义。

Url编码只是在简单的在各个特殊的字符串前加上%,然后进行ASCII编码，例如，对上述产生歧义的字符串进行Url编码后的结果是："name1=va%26lu%3D",这样服务端会把紧跟在"%"后的字节当成普通的字节，就是不会把它当成各个参数或者是键值对的分隔符。

通常，开发中如果某一个参数需要编码，说明这个参数不适合传输，原因多种多样，例如，size过大，包含隐私数据，对于Url之所以要编码,是因为Url中存在引起服务端解析歧义的特俗参数。

例如，Url参数字符串中使用key=value键值对这样的形式来传参，键值对之间以&符号分割，如：?name=adb&address=efg。如果你的value中包含了&或者=，那么势必会造成接收Url的服务器解析错误，因此必须将引起歧义的&和=符号进行转义，也就是对其进行编码。

Url的编码格式采用的是ASCII,不是Unicode，这也就说明你不能在Url中包含任何非ASCII字符，例如：中文，否则如果客户端浏览器和服务端浏览器支持的字符集不同的情况下，中文可能会造成问题。
注意：浏览器会在传输过程中内部进行对中文进行编码，到达服务端时Tomcat自动解码。

Url编码的原则就是使用安全的字符（没有特殊用途或者特殊意义的可打印字符）去表示那些不安全的字符。

一、哪些字符需要编码

RFC3986文档规定，Url中只允许包含英文字母（a-zA-Z）、数字（0-9）、-_.~4个特殊字符以及所有保留字符。RFC3986文档对Url的编解码问题做出了详细的建议，指出了哪些字符需要被编码才不会引起Url语义的转变，以及对为什么这些字符需要编码做出了相应的解释。

US-ASCII字符集中没有对应的可打印字符：Url中只允许使用可打印字符。US-ASCII码中的10-7F字节全都表示控制字符，这些字符都不能直接出现在Url中。同时，对于80-FF字节（ISO-8859-1），由于已经超出了US-ACII定义的字节范围，因此也不可以放在Url中。

保留字符：Url可以划分成若干个组件，协议、主机、路径等。有一些字符 是用作分隔不同组件的。
例如：冒号是用于分隔协议和主机， / 斜杠是用于分隔主机和路径，？ 问号是用于分隔路径和参数，等等，还有有一些字符串（!$&'()*+,;=）用于在每个组件中起到分隔作用的，如=用于表示查询参数中的键值对，&符号用于分隔查询多个键值对。当组件中的普通数据包含这些特殊字符时，需要对其进行编码。
RFC3986中指定了以下字符为保留字符：! * ' ( ) ; : @ & = + $, / ? # [ ]
不安全字符：还有一些字符，当他们直接放在Url中的时候，可能会引起解析程序的歧义，这些字符被视为不安全字符，原因有很多。

> 空格：Url在传输过程中，或者用户在排版的过程。或者文本处理程序在处理文本时，都会引入一些无关紧要的空格，或者将那些有意义的空格去掉。
>
> “” 引号以及<> 尖括号：引号和尖括号通常用于在普通文本中起到分隔Url的作用
>
> #：通常用于表示书签或者锚点
>
> %：百分号本身用作对不安全字符进行编码时使用的特殊字符，因此本身需要编码。
>
> {}|^[]`~：某一些网关或者传输代理会篡改这些字符
> 需要注意的是，对于Url中的合法字符，编码和不编码都是等价的，但是对于上面提到的这些字符，如果不经过编码，那么他们有可能会造成Url语义的不同。因此对于Url而言，只有普通英文字符和数字，特殊字符$-_.+!*'()还有保留字符，才能出现在未经编码的Url之中。其它字符均需要经过编码之后才能出现在Url中。
>
> 但是由于历史原因，目前尚存在一些不标准的编码实现，例如对于~ 波浪符号，不需要进行Url编码，但是还是有很多老的网关或者传输代理会进行编码。

二、如何对Url中的非法字符进行编码

Url编码通常也被称为百分号编码（Url Encoding，also known as percent-encoding），是因为它的编码方式非常简单，**使用%百分号加上两位的字符——0123456789ABCDEF——代表一个字节的十六进制形式**。Url编码使用的默认字符集是US-ASCII。例如：a在US-ASCII码中对应的字节是0x61,那么Url编码之后得到的就是%61，我们在地址栏输入http://g.cn/search?q=%61%62%63，实际上就等同于在google上搜索abc了。又如@符号在ASCII字符集中对应的字节为0x40，经过Url编码之后得到的是%40。
对于非ASCII字符，需要使用ASCII字符集的超集进行编码得到相应的字节，然后对每个字节执行百分号编码。对于Unicode字符，RFC文档建议使用utf-8对其进行编码得到相应的字节，然后对每个字节执行百分号编码。如"中文"使用UTF-8字符集得到的字节为0xE4 0xB8 0xAD 0xE6 0x96 0x87，经过Url编码之后得到"%E4%B8%AD%E6%96%87"。

如果某个字节对应着ASCII字符集中的某个非保留字符，则此字节无需使用百分号表示。例如："Url编码",使用UTF-8编码得到字节是0x55 0x72 0x6C 0xE7 0xBC 0x96 0xE7 0xA0 0x81，由于前三个字节对应着ASCII中的非保留字符"Url"，因此这三个字节可以用非保留字符"Url"表示。最终的Url编码可以简化成"Url%E7%BC%96%E7%A0%81"，当然，如果你用"%55%72%6C%E7%BC%96%E7%A0%81"也是可以的。
由于历史的原因，有一些Url编码实现并不完全遵循这样的原则，下面会提到。

三、Javascript中的escape, encodeURI和encodeURIComponent的区别

JavaScript中提供了3对函数用来对Url编码以得到合法Url,它们分别是escape / unescape, encodeURI / decodeURI和encodeURIComponent / decodeURIComponent。由于解码和编码的过程是可逆的，因此这里只解释编码的过程。
这三个编码的函数——escape，encodeURI，encodeURIComponent——都是用于将不安全不合法的Url字符转换为合法的Url字符表示，它们有以下几个不同点。

安全字符不同：

下面列出了这三个函数的安全字符（即函数不会对这些字符进行编码）

- escape（69个）：*/@±._0-9a-zA-Z
- encodeURI（82个）：!#$&'()*+,/:;=?@-._~0-9a-zA-Z
- encodeURIComponent（71个）：!'()*-._~0-9a-zA-Z

兼容性不同：

escape函数是从Javascript 1.0的时候就存在了，其他两个函数是在Javascript 1.5才引入的。但是由于Javascript 1.5已经非常普及了，所以实际上使用encodeURI和encodeURIComponent并不会有什么兼容性问题。

对Unicode字符的编码方式不同：

这三个函数对于ASCII字符的编码方式相同，均是使用百分号+两位十六进制字符来表示。但是对于Unicode字符，escape的编码方式是%uxxxx，其中的xxxx是用来表示unicode字符的4位十六进制字符。这种方式已经被W3C废弃了。但是在ECMA-262标准中仍然保留着escape的这种编码语法。encodeURI和encodeURIComponent则使用UTF-8对非ASCII字符进行编码，然后再进行百分号编码。这是RFC推荐的。因此建议尽可能的使用这两个函数替代escape进行编码。

适用场合不同：

encodeURI被用作对一个完整的URI进行编码，而encodeURIComponent被用作对URI的一个组件进行编码。

从上面提到的安全字符范围表格来看，我们会发现，encodeURIComponent编码的字符范围要比encodeURI的大。
我们上面提到过，保留字符一般是用来分隔URI组件（一个URI可以被切割成多个组件，参考预备知识一节）或者子组件（如URI中查询参数的分隔符），如：号用于分隔scheme和主机，?号用于分隔主机和路径。由于encodeURI操纵的对象是一个完整的的URI，这些字符在URI中本来就有特殊用途，因此这些保留字符不会被encodeURI编码，否则意义就变了。

组件内部有自己的数据表示格式，但是这些数据内部不能包含有分隔组件的保留字符，否则就会导致整个URI中组件的分隔混乱。因此对于单个组件使用encodeURIComponent，需要编码的字符就更多了

Java后台解析编码方式

如果JS通过encodeUrl和encodeUrlCompent编码，服务端如何解码得到正确的中文

```
方法1:
//%形式的中文被服务器自动iso解码后就会乱码，所以先要用 iso8859-1解码后获取到utf8编码值，然后再解码
String aaa = URLDecoder.decode(URLEncoder.encode(aaa, CommonUtils.ISO8859_1), CommonUtils.UTF8);
方法2:
String aaa = new String(aaa.getBytes(CommonUtils.ISO8859_1), CommonUtils.UTF8);
```

四、表单提交

当Html的表单被提交时，每个表单域都会被Url编码之后才被发送。由于历史原因，表单提交使用的Url编码实现并不符合最新的标准。例如对于空格使用的编码并不是%20，而是+号，如果表单使用的提交方法是Post,我们可以在HTTP请求头中看到有一个Content-Type的header，值为application/x-www-form-unlencoded。大部分应用程序均能处理这种非标准实现的Url编码，但是在客户端Javascript中，并没有一个函数能够将+号解码成空格，只能自己写转换函数。还有，对于非ASCII字符，使用的编码字符集取决于当前文档使用的字符集。例如我们在Html头部加上

```
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
```

这样浏览器就会使用gb2312去渲染此文档（注意，当HTML文档中没有设置此meta标签，则浏览器会根据当前用户喜好去自动选择字符集，用户也可以强制当前网站使用某个指定的字符集）。当提交表单时，Url编码使用的字符集就是gb2312。

五、 生产环境灾难

前端特殊空格ASCII码为160，后端无法去除

在一个文本框中前端提交，出现了一个不换行的空格，后端用很多方法都无法去除此空格。
不间断空格有个问题，就是它无法被trim()所裁剪，也无法被正则表达式的\s所匹配，也无法被StringUtils的isBlank()所识别，也就是说，无法像裁剪寻常空格那样移除这个不间断空格。
表象方法：可以利用不间断空格的Unicode编码来移除它，其编码为\u00A0。

```
replace("\u00A0", "")
replaceAll("\\u00A0+", "")  //这是正则表达式的写法

String str = "aacsdfe ";  //包含了不间断空格的字符串
str = str.replace("\u00A0", "");
str = str.replaceAll("\\u00A0+", "");
```

可是为什么会产生一个 不间断的空格呢？为什么其它功能的输入框就没有出现。判断是前端的问题，问题成功推锅，转移到前端身上，哈哈哈。
经过前端排查，错误的样式编写，传到后端会出现一个不间断的空格。

```
</div>
<div class="p12 g-white flexbox je-align-center jepor jecell-top">
    <div class="flexbox jeflex edibox">
        <!-- <div contenteditable="true" class="pt11 pb11 f13 remark-input" id="remarkInput"></div> -->
```

参考链接：
1、一些http或https请求的参数，什么情况下需要urlencode编码
2、殊的空格-ASCII码值160

真香号
码龄5年 企业员工
231 原创 | 4万+ 周排名 | 2万+ 总排名 | 38万+ 访问 | 等级
4842 积分 | 139 粉丝 | 242 获赞 | 69 评论 | 836 收藏
私信 关注

强烈不推荐 不推荐 一般般 推荐 强烈推荐

最新文章

（一）什么是ElasticSearch

从餐厅服务员到一线电商程序员(中)

JVM 初探 (三)：方法区、栈区

2022年 11篇　2021年 3篇
2020年 85篇　2019年 57篇
2018年 82篇

python爬取内容乱码_python爬取html中文乱码　weixin_39756273的博客　47
环境：python3.6爬取代码：import requestsurl = 'https://www.dygod.net/html/tv/hytv/'req = requests.get(url)print(req.text)爬取结果：µçÊÓ¾ç / »ªÓïµçÊ…

HTTP之URLEncode加密的请求数据获取及转json　人真正变得强大，不是因为守护着自尊心，而是抛开自尊心的时候。　388
// 请求数据获取 public static String getPostString(HttpServletRequest request) { BufferedReader in = null; String parameters = ""; try { in = new Buffere…

“相关推荐”对你有帮助么？

非常没帮助　没帮助　一般　有帮助　非常有帮助

真香号　关注

2　2　0　专栏目录

举报