# getDisplayCountry

**public final** String getDisplayCountry() {  
 **return** getDisplayCountry(getDefault(Category.DISPLAY));  
}

getDisplayCountry方法用于获得适合向用户显示的语言环境国家/地区名。

语法  public final String getDisplayCountry()

返回值：返回适合向用户显示的语言环境国家/地区名。

**public** String getDisplayCountry(Locale locale) {  
 String countryCode = baseLocale.getRegion();  
 **if** (countryCode.isEmpty()) {  
 **return ""**;  
 }  
  
 **final** String normalizedRegion = normalizeAndValidateRegion(  
 countryCode, **false** */\* strict \*/*);  
 **if** (normalizedRegion.isEmpty()) {  
 **return** countryCode;  
 }  
  
 String result = ICU.getDisplayCountry(**this**, locale);  
 **if** (result == **null**) { *// TODO: do we need to do this, or does ICU do it for us?* result = ICU.getDisplayCountry(**this**, Locale.getDefault());  
 }  
 **return** result;  
}

## ICU.getDisplayCountry

**public static** String getDisplayCountry(Locale targetLocale, Locale locale) {  
 **return** getDisplayCountryNative(targetLocale.toLanguageTag(), locale.toLanguageTag());  
}

**private static native** String getDisplayCountryNative(String targetLanguageTag, String languageTag);

## ICU\_getDisplayCountryNative

**static** jstring ICU\_getDisplayCountryNative(JNIEnv\* env, jclass, jstring javaTargetLanguageTag, jstring javaLanguageTag) {  
 ScopedIcuLocale icuLocale(env, javaLanguageTag);  
 **if** (!icuLocale.valid()) {  
 **return** NULL;  
 }  
 ScopedIcuLocale icuTargetLocale(env, javaTargetLanguageTag);  
 **if** (!icuTargetLocale.valid()) {  
 **return** NULL;  
 }  
  
 icu::UnicodeString str;  
 icuTargetLocale.locale().getDisplayCountry(icuLocale.locale(), str);  
 **return** env->NewString(str.getBuffer(), str.length());  
}

## Locale::getDisplayCountry

// external/icu/icu4c/source/common/locdispnames.cpp

nicodeString&  
Locale::getDisplayCountry(**const** Locale &displayLocale,  
 UnicodeString &result) **const** {  
 UChar \*buffer;  
 UErrorCode errorCode=U\_ZERO\_ERROR;  
 int32\_t length;  
  
 buffer=result.getBuffer(ULOC\_FULLNAME\_CAPACITY);  
 **if**(buffer==0) {  
 result.truncate(0);  
 **return** result;  
 }  
  
 length=uloc\_getDisplayCountry(fullName, displayLocale.fullName,  
 buffer, result.getCapacity(),  
 &errorCode);  
 result.releaseBuffer(U\_SUCCESS(errorCode) ? length : 0);  
  
 **if**(errorCode==U\_BUFFER\_OVERFLOW\_ERROR) {  
 buffer=result.getBuffer(length);  
 **if**(buffer==0) {  
 result.truncate(0);  
 **return** result;  
 }  
 errorCode=U\_ZERO\_ERROR;  
 length=uloc\_getDisplayCountry(fullName, displayLocale.fullName,  
 buffer, result.getCapacity(),  
 &errorCode);  
 result.releaseBuffer(U\_SUCCESS(errorCode) ? length : 0);  
 }  
  
 **return** result;  
}

## uloc\_getDisplayCountry

U\_CAPI int32\_t U\_EXPORT2  
uloc\_getDisplayCountry(**const char** \*locale,  
 **const char** \*displayLocale,  
 UChar \*dest, int32\_t destCapacity,  
 UErrorCode \*pErrorCode) {  
 **return** \_getDisplayNameForComponent(locale, displayLocale, dest, destCapacity,  
 uloc\_getCountry, \_kCountries, pErrorCode);  
}

## \_getDisplayNameForComponent

U\_ICUDATA\_REGION=**"icudt" "58" "x" "-" "region"= icudt58x-region**

#define U\_ICUDATA\_REGION U\_ICUDATA\_NAME U\_TREE\_SEPARATOR\_STRING **"region"**

#define U\_ICUDATA\_LANG U\_ICUDATA\_NAME U\_TREE\_SEPARATOR\_STRING **"lang"**

#define U\_TREE\_SEPARATOR\_STRING **"-"**

#define U\_ICUDATA\_NAME **"icudt"** U\_ICU\_VERSION\_SHORT U\_ICUDATA\_TYPE\_LETTER

#define U\_ICU\_VERSION\_SHORT **"58"**

#if U\_CHARSET\_FAMILY  
# if U\_IS\_BIG\_ENDIAN  
 */\* EBCDIC - should always be BE \*/*# define U\_ICUDATA\_TYPE\_LETTER **"e"**# define U\_ICUDATA\_TYPE\_LITLETTER e  
# else  
# error **"Don't know what to do with little endian EBCDIC!"**# define U\_ICUDATA\_TYPE\_LETTER **"x"**# define U\_ICUDATA\_TYPE\_LITLETTER x  
# endif  
#else  
# if U\_IS\_BIG\_ENDIAN  
 */\* Big-endian ASCII \*/*# define U\_ICUDATA\_TYPE\_LETTER **"b"**# define U\_ICUDATA\_TYPE\_LITLETTER b  
# else  
 */\* Little-endian ASCII \*/*# define U\_ICUDATA\_TYPE\_LETTER **"l"**# define U\_ICUDATA\_TYPE\_LITLETTER l  
# endif  
#endif

**static** int32\_t  
\_getDisplayNameForComponent(**const char** \*locale,  
 **const char** \*displayLocale,  
 UChar \*dest, int32\_t destCapacity,  
 UDisplayNameGetter \*getter,  
 **const char** \*tag,  
 UErrorCode \*pErrorCode) {  
 **char** localeBuffer[ULOC\_FULLNAME\_CAPACITY\*4];  
 int32\_t length;  
 UErrorCode localStatus;  
 **const char**\* root = NULL;  
  
 */\* argument checking \*/* **if**(pErrorCode==NULL || U\_FAILURE(\*pErrorCode)) {  
 **return** 0;  
 }  
  
 **if**(destCapacity<0 || (destCapacity>0 && dest==NULL)) {  
 \*pErrorCode=U\_ILLEGAL\_ARGUMENT\_ERROR;  
 **return** 0;  
 }  
  
 localStatus = U\_ZERO\_ERROR;  
 length=(\*getter)(locale, localeBuffer, **sizeof**(localeBuffer), &localStatus);  
 **if**(U\_FAILURE(localStatus) || localStatus==U\_STRING\_NOT\_TERMINATED\_WARNING) {  
 \*pErrorCode=U\_ILLEGAL\_ARGUMENT\_ERROR;  
 **return** 0;  
 }  
 **if**(length==0) {  
 **return** u\_terminateUChars(dest, destCapacity, 0, pErrorCode);  
 }  
  
 root = tag == \_kCountries ? U\_ICUDATA\_REGION : U\_ICUDATA\_LANG;  
  
 **return** \_getStringOrCopyKey(root, displayLocale,  
 tag, NULL, localeBuffer,  
 localeBuffer,  
 dest, destCapacity,  
 pErrorCode);  
}

## \_getStringOrCopyKey

**static** int32\_t  
\_getStringOrCopyKey(**const char** \*path, **const char** \*locale,  
 **const char** \*tableKey,   
 **const char**\* subTableKey,  
 **const char** \*itemKey,  
 **const char** \*substitute,  
 UChar \*dest, int32\_t destCapacity,  
 UErrorCode \*pErrorCode) {  
 **const** UChar \*s = NULL;  
 int32\_t length = 0;  
  
 **if**(itemKey==NULL) {  
 */\* top-level item: normal resource bundle access \*/* UResourceBundle \*rb;  
  
 rb=ures\_open(path, locale, pErrorCode);  
  
 **if**(U\_SUCCESS(\*pErrorCode)) {  
 s=ures\_getStringByKey(rb, tableKey, &length, pErrorCode);  
 */\* see comment about closing rb near "return item;" in \_res\_getTableStringWithFallback() \*/* ures\_close(rb);  
 }  
 } **else** {  
 */\* Language code should not be a number. If it is, set the error code. \*/* **if** (!uprv\_strncmp(tableKey, **"Languages"**, 9) && uprv\_strtol(itemKey, NULL, 10)) {  
 \*pErrorCode = U\_MISSING\_RESOURCE\_ERROR;  
 } **else** {  
 */\* second-level item, use special fallback \*/* s=uloc\_getTableStringWithFallback(path, locale,  
 tableKey,   
 subTableKey,  
 itemKey,  
 &length,  
 pErrorCode);  
 }  
 }  
  
 **if**(U\_SUCCESS(\*pErrorCode)) {  
 int32\_t copyLength=uprv\_min(length, destCapacity);  
 **if**(copyLength>0 && s != NULL) {  
 u\_memcpy(dest, s, copyLength);  
 }  
 } **else** {  
 */\* no string from a resource bundle: convert the substitute \*/* length=(int32\_t)uprv\_strlen(substitute);  
 u\_charsToUChars(substitute, dest, uprv\_min(length, destCapacity));  
 \*pErrorCode=U\_USING\_DEFAULT\_WARNING;  
 }  
  
 **return** u\_terminateUChars(dest, destCapacity, length, pErrorCode);  
}

# external/icu/icu4c/source/data

/region/af.txt

# 编译external/icu4c/data方法

Android 7.1 icu4c(icudt56l.dat)

https://blog.csdn.net/yin1031468524/article/details/74276633

<https://blog.csdn.net/andy_android/article/details/20550893?utm_source=blogxgwz9>

bullhead:/system/usr/icu # ls

icudt58l.dat

adb pull /system/usr/icu/icudt58l.dat