

From Characters to Web Services .. to Internationalization is Everywhere

Felix Sasaki

Introduction

This article has two purposes. First, it shows that internationalization is relevant for a wide range of technologies: character encoding, bidirectional text, text formatting, character processing, datatype specific issues etc. The article gives a brief overview of some of these technologies, without being exhaustive. Second, the article discusses chances and challenges for possibly new areas of internationalization like internationalized web services or markup for internationalization purposes. The conclusion emerging from these discussions is that the nature of "internationalization" depends on the technology in question. For each technology there are new chances and challenges for internationalization.

What is Internationalization?

"Internationalization" (I18N) is the process of making a product or its underlying technology ready for applications in various languages, cultures and regions. The acronym of Internationalization is used as "I18N" because there are eighteen characters or letters in between the first letter 'I' and the last letter 'N'. Closely related to internationalization is "localization" (L10N), which is the process of adapting the product or technology to a "locale" (a specific language, country, or market). The acronym of Localization is used as "L10N" because there are ten characters or letters in between the first letter 'L' and the last letter 'N'. An example which demonstrates the relation between internationalization and localization is the Java class "resourceBundle". It allows the user to gather locale information. It is an internationalization feature of the Java language that it provides this class. The localization process means to use a "resourceBundle" e.g. for the adaptation of a user interface to a locale. The World Wide Web Consortium (or W3C) Internationalization Activity has the goal of proposing and coordinating any techniques, conventions, guidelines and activities within the W3C and together with other organizations that allow and make it easy to use W3C technology worldwide, with different languages, scripts, and cultures.

Characters

The topic of characters is at the center of many internationalization efforts. The Java class "resourceBundle" would be useless without the ability of Java to process a wide range of characters, e. g. to create user interfaces for people all across the world. Java makes use of the Unicode [1] standard; this provides a large character repertoire which is constantly extended. The encoding of characters [2] encompasses the following steps: 1) Identification of a character repertoire. In Unicode, characters are organized within scripts. For example, many of the language or region specific differences between ideographic characters in China, Japan and Korea are unified into a single script.

- 2) Assignment of numeric identifiers for characters (code points, character numbers, code positions).
- 3) Choice of a base datatype like byte.
- 4) Choice of a serialization scheme for the transmission of the datatype units. This encompasses the choice of the byte-order for multi-byte base datatypes.

The result of step 4), together with the character repertoire defined in 1), is called a "character encoding". IANA charset identifiers [3] provide unique names for character encodings. Unicode has different "Unicode encoding forms" like UTF-8, UTF-16 and UTF-32. All Unicode encoding forms share the same character repertoire, but have different base datatypes and serialization schemes.

Text 1: Bidirectional Text

Going from the level of characters to the level of text, there are internationalization specific features as well. Some scripts like Hebrew or Arabic are visualized from the right-to-left. It is common to store them in memory the way they are typed (the so called "logical order", opposed to the "visual order"). To support appropriate visualization, each character in Unicode has a property "Bidi_Class". The Unicode Bidirectional Algorithm [4] uses this property for the positioning of characters. But there are cases when the algorithm needs helps. The following example is based on HTML 4.01 [5]:

Source code: english1 [HEBREW2 english3 HEBREW4] english5
 Visualization a): english1 "2WERBEH english3 4WERBEH" english5
 Visualization b): english1 "4WERBEH english3 2WERBEH" english5

The source code is an English text with a Hebrew citation, which itself contains an English citation. That is, the overall "directional run" is left-to-right, with an embedded right-to-left sequence, which contains a left-to-right sequence. Applying only the Bidirectional Algorithm, the visualization would be a). But the appropriate visualization for the Hebrew citation with the embedded English citation would be b). To achieve b), Unicode code points (here marked with "*U+*") can be used to signal the beginning and the end of a new directional "embedding level" for the Hebrew text: the character "RIGHT-TO-LEFT EMBEDDING" (U+202B), and the character "POP DIRECTIONAL FORMATTING" (U+202C):

english1 [*U+202B*HEBREW2 english3 HEBREW4*U+202C*] english5

In HTML, a different method is available and recommended: a "dir" attribute should be used to specify the new embedding level for the Hebrew citation, and no Unicode characters:

english1 [HEBREW2 english3 HEBREW4] english5

There are many other aspects of the Bidirectional Algorithm. A good overview is given by [6].

Text 2: Text Formatting

Besides bidirectionality, "ruby" is another aspect of internationalization issues in text. "Ruby" is a term used for a run of annotation text that is associated with another run of text, referred to as the base text. It is used mainly in East Asian scripts, for example to provide a reading (pronunciation) guide. The ruby specification [7] defines markup, which can be used to specify the base text and to add a ruby text:

```
<ruby>  
<rb>宮崎駿</rb>  
<rt>みやざきはやお</rt>  
</ruby>
```

[7] is only concerned with the structure of ruby markup. CSS ("Cascading Style Sheets") properties which can be used for the rendering of ruby are being defined in [8].

Operations on Characters: Counting, Comparison and Indexing

The Bidirectional Algorithm and ruby are mainly concerned with the visualization of text. Nevertheless, internationalization also heavily comes into play during the processing of characters. A basic process is counting characters. Here are areas one has to be careful about. For example, regular expressions in Java count character borders. That is, Java takes the beginning of an input sequence into account, even if it is empty. On the other hand, XML Schema regular expressions count "only" characters. Hence, given the empty input sequence "" and the regular expression "a?", there will be a match in Java, but not in XML Schema.

Another important operation is the comparison or indexing of strings. There are two prerequisites for comparison. First, the strings have to be in the same encoding (which is not trivial to assure in the World Wide Web), and second, they have to be in the same "normalization form". Normalization is the process of bringing two strings to a canonical encoding before they are processed. This is necessary because some character encodings allow multiple representations for the same string. An example: "Á" can be represented in Unicode as a single character with the code point U+00E7 "Á" or as a sequence U+0063 "c" U+0327 "_". Normalization of text and string identify matching is described in detail in the "Character Model for the World Wide Web 1.0: Normalization" [10].

A "collation" is a specification of the manner in which character strings are compared and ordered. A simple collation is a code point based collation. It is used for example as the default collation in XPath 2.0 functions [9]. "Code point based" means that strings are compared via the numeric identifiers of code points. More enhanced collations take locale specific information into account. For example, a collation might

identify the two strings "Strasse" and "Stra_e" as identical or different, depending on the underlying locale.

Datatype-specific Operations: Processing Time Information

The following examples for internationalization issues leave the area of characters and textual data. They are specific to datatypes used for handling time related information. [11] lists various kinds of time information:

- Incremental time, as for example in the Java class "java.util.date". Incremental time starts at an "epoch" and typically counts time in milliseconds. It is used for instance in time stamps.
- Field based time. Time information is separated into fields like minute, hour, month or year. There are two variants: time zone independent versus time zone dependent field based time. The former is used e. g. for a holiday schedules, the latter for purchase order dates.

The notion of a "time zone" is sometimes confusing. It is related to the concept of a "zone offset", but not quite the same. A zone offset is the difference of a time value to UTC ("Universal Coordinated Time"). The zone offset can change for the same region e. g. due to daylight savings time. A time zone then identifies a region, taking the offset to UTC and such changes into account. The "Olson time zone database" [12] provides data about regions, offset changes and the time zone identifiers respectively.

The topic gets even more complicated as it comes to implementations of time information. Java uses incremental time. On the other hand, SQL uses field based time. A source for confusion can be that a "java.util.date" object provides still an incremental value, even if it is created from an SQL data base.

As for XML Schema, the situation is different again: XML Schema datatypes like "time" [13] are defined as field based values with an optional parameter "time zone". But this parameter is actually a zone offset, which does not make use of information about regions and daylight savings time. Since the parameter is optional, a data set can encompass values with and without offset information, which then cannot be put into a total ordering. XPath 2.0 functions - which rely on XML Schema - provide an implicit time zone for ordering operations, which is defined by the implementation of XPath. To avoid a dependency on implementations, it is recommended to set UTC as the implicit time zone.

Locale in Web Services - and More

Starting from characters, going from their visualization and their processing to datatype specific issues, web services are another technology area which can benefit from internationalization. The general goal of web services is to be independent of operating systems and programming languages. Sometimes a web service makes use of information about user locale preferences. The Accept-language header sent by a user agent may provide some information for a web server to infer the locale of the user. But this might not be enough to infer various kinds of locale specific information, the time zone and other international preferences like currencies or units of measurement. Hence, the goal

of "Web Services Internationalization (WS-I18N)" [14] is to provide capabilities in web services descriptions (WSDL) and SOAP messages to convey this kind of information.

One challenge for web services internationalization is the task of identifying a language or locale. This is difficult because the current standard for language information, RFC 3066, has deficits for example with respect to the stability of language tags. Its recently approved successor [15] provides

- a grammar for defining subtags for language, scripts, regions, variants etc.
- a publicly accessible registry for subtags.
- a matching scheme for language tag values.

This will be the base for reliable processing of language information.

The Present and the Future: Markup for Internationalization and Localization

As for internationalization issues in markup languages, there are two areas to be taken into account. One aspect is the question of which characters are useful or not useful for an application within markup. This topic is discussed in [16]. Another aspect encompasses internationalization facilities of the markup itself. An example here is the "ltr" attribute in HTML, which has been discussed in this article. It would be useful to have such facilities for many markup schemes, without "reinventing the wheel" again and again. Hence, a goal is to have a single set of elements and attributes, which fulfills internationalization requirements. Here also localization issues come into play again, and they are strongly related to internationalization. An example for localization relevant markup, which should be applicable for many markup schemes, is an attribute "translate". Its purpose is to identify translatable and non-translatable text. This is an area of current development. Requirements for the internationalization and localization of markup are formulated in [17]; a first current implementation of the envisaged tag set is discussed in [18].

A topic yet to be covered in a future version of this tag set is markup to express syntactic, semantic and other linguistic information in documents [19]. Machine translation tools can highly benefit from such markup. This is another area of technology, which demonstrates that the topic of internationalization has still not come to an end.

References

- [1] <http://www.unicode.org>
- [2] <http://www.w3.org/TR/2005/REC-charmod-20050215/#sec-Characters>
- [3] <http://www.iana.org/assignments/character-sets>
- [4] <http://www.unicode.org/reports/tr9/tr9-15.html>
- [5] <http://www.w3.org/TR/html40/struct/dirlang.html#h-8.2>
- [6] <http://www.w3.org/International/articles/inline-bidi-markup/>
- [7] <http://www.w3.org/TR/2001/REC-ruby-20010531/>
- [8] <http://www.w3.org/TR/2003/CR-css3-ruby-20030514>
- [9] <http://www.w3.org/TR/2005/CR-xpath-functions-20051103/#collations>
- [10] <http://www.w3.org/TR/2005/WD-charmod-norm-20051027/>
- [11] <http://www.w3.org/tr/2005/NOTE-timezone-20051013/>

- [12] <http://www.twinsun.com/tz/tz-link.htm>
- [13] <http://www.w3.org/TR/xmlschema-2/#time>
- [14] <http://www.w3.org/TR/2005/WD-ws-i18n-20050914/>
- [15] <http://www.ietf.org/internet-drafts/draft-ietf-ltru-registry-14.txt>
- [16] <http://www.unicode.org/reports/tr20/tr20-7.html>
- [17] <http://www.w3.org/TR/2005/WD-itsreq-20051122/>
- [18] <http://www.w3.org/TR/2005/WD-its-20051122/>
- [19] <http://esw.w3.org/topic/its0908LinguisticMarkup>

Author's Biography:

Felix Sasaki has joined the W3C in April 2005 to work in the Internationalization Activity. He is part of the team at Keio-SFC. His main field of interest is the combined application of W3C technologies for representation and processing of multilingual information. From 1993 until 1999, Felix studied Japanese and Linguistics in Berlin, Nagoya (Japan) and Tokyo. Since 1999 he worked in the Department of Computational Linguistics and Text-technology, at the University of Bielefeld (Germany), where he finished his PhD in 2004. The PhD deals with the integration of heterogeneous linguistic resources using XML-based (e.g. linguistic corpora) and RDF-based (e.g. lexicon, conceptual models) representations. His hobbies - except playing with his children - are reading and Karaoke. Email: fsasaki@w3.org. Post: Keio Research Institute at SFC, 5322 Endo Fujisawa, Kanagawa, 252-8520 Japan. Tel: +81.466.49.1170. Fax: +81.466.49.1171.