Integrating internationalized websites with databases and email systems: working with multilingual texts

Fitzgerald, W.

Report (National Research Council Canada. Radio and Electrical Engineering Division : ERB; no. ERB-1107, 2004-04

National Research Council Canada
Conseil national de recherches Canada

Institute for Information Technology
Institut de technologie de l'information

NRC · CNRC

Integrating Internationalized Websites with Databases and Email Systems: Working with Multilingual Texts *

Fitzgerald, W.
April 2004

* published as NRC/ERB-1107. 14 pages. April 6, 2004. NRC 46562.

Canada

National Research Council Canada
Conseil national de recherches Canada

ERB-1107

Institute for Information Technology
Institut de technologie de l'information

NRC-CNRC

Integrating Internationalized Websites with Databases and Email Systems: Working with Multilingual Texts

Fitzgerald, W.
April 2004

Canada

NRC 46562

Integrating internationalized websites with databases and email systems: Working with multilingual texts

Will Fitzgerald
National Research Council Canada
will.fitzgerald@nrc-cnrc.gc.ca

Introduction

It is increasingly important to build websites that easily reach an international audience. At the same time, it is increasingly important for Web pages to not only present static, non-changing pages, but also to provide websites that interact with other systems, such as databases and electronic mail. This white paper provides various tips, tricks and some best practices for building an integrated, internationalized web application.

An example: Conducting polls on the multilingual Web

International Polling[1] conducts marketing polls in a wide variety of countries and languages. Their clients deploy polls on the Web and receive the results via email or the Web. *International Polling* staff manage the creation of polling web pages, data collection, and reporting. The basic workflow is something like this: The client (via email, documents or the telephone) indicates the content of the poll they would like to take, as well as a list of the names, email addresses and preferred language of poll respondents. *International Polling* staff then create the poll, which can be conducted in any of the specified languages. Email messages are then sent to each of the respondents (in the respondent's preferred language) telling them how to access the poll website. Respondents have a time window in which to take the poll. The client can log into the *International Polling* website to monitor progress and receive poll results.

Technology used

International Polling publishes their web polling system using the Apache web server. *International Polling* staff create poll contents by using the web, using standard HTML forms. The content is placed into a database that runs on the PostgreSQL database

[1] Not a real company, of course.

system, and a Java program creates static pages for each language of each poll. Another Java program initiates sending the email to respondents, telling them how to log onto the system. When respondents take the poll, their data (both textual and numeric) are placed in the database. Although we will not discuss it here, yet a third Java program is required for providing the web-based reports to the client.

Example poll

The *Conseil international de producteurs de fromage* (International Council of Cheese Producers[2]) has engaged *International Polling* to ask people in 3 countries such questions as: *How much cheese have you purchased in the last 24 hours?* and *What do you especially like about cheese?* The countries are Canada (where the survey must be available in French or English), Russia, and the People's Republic of China.

French	English
Conseil international des producteurs de fromage	International Council of Cheese Producers
1. Quelle quantité de fromage avez-vous acheté dans les dernières 24 heures? 2. Qu'est-ce que vous aimez du fromage?	1. How much cheese have you purchased in the last 24 hours? 2. What do you like about cheese?
Chinese	Russian
国际乳酪厂商理事会	Международный комитет производителей сыра
1. 您在过去24小时内购买了多少乳酪? 2. 您喜欢什么样的乳酪?	1. Сколько сыра вы купили за последние 24 часа? 2. Чем вам нравится сыр?

Table 1. Example questions in the International Cheese Poll

Basics of text internationalization

There are basically four things you have to know about text and text internationalization. First, you have to know how individual characters—like 'a' or '$' or 'é' or even '麼' or 'あ'--are mapped to numbers, or their *encoding*. (The number associated with a character is called its *code point*; an encoding is a set of character to code point associations). Secondly, you need to know how these mappings can be

[2] Also, of course, not a real organization.

encoded into the bits that computers understand; their *encoding scheme*[3]. Third, you need to know how these characters can be displayed—their *font*. And fourth, you need to know how to tell different applications about which encodings and fonts should be used.

Computers, of course, are primarily number crunching machines, and this is why encoding and encoding schemes are so important. Perhaps the most widespread encoding is ASCII[4]. In ASCII, the letter 'a' is assigned the number 97, the letter 'A' is number 41 (see Appendix 1, which show the mappings from 32 to 127).

So prevalent is ASCII that for many computer systems and applications, text is assumed to be in ASCII format unless otherwise specified. Although it is nice to have this near guarantee, (an 'a' is almost always represented by the number 92, etc.), the obvious problem is how to encode characters such as 'é,' '麼' and 'あ' (to give just three examples). The simple answer is just this: ASCII cannot do it. There are only 128 ASCII numbers (or just 96 if you ignore the control codes from 0 to 31), and this cannot express the richness of all the world's characters.

128 numbers require seven bits to represent; add another bit—eight bits or a standard *byte*[5]—and you can represent 256 numbers/characters. The typical trick has been to create encodings that retain the ASCII characters, but map different sets of characters to 129 and above. So, for example, the encoding called ISO-8859-15[6] (or "Latin 9") assigns the character 'é' to 233, and is sufficient for the Western languages, broadly speaking. In other words, for our example, if *International Poll* determined that the respondents would only speak the most common languages that come out of Europe (English, Spanish, French, etc.) they could use the Latin 9 encoding without problem[7].

---

[3] Often the encoding and the encoding scheme are conflated; as we'll see, ISO-8859-9 is an encoding for characters in Western languages with an 8-bit encoding scheme; this is often called an encoding or *character set*.

[4] The American Standard Code for Information Interchange.

[5] 8 bits is also called an *octet*.

[6] Created by the International Standards Organization.

[7] ISO-8859-15 is based on ISO-8859-1 ("Latin 1"). It is essentially the same as Latin 1, with the addition of the Euro sign ('€'). Latin 1 was the default encoding scheme for earlier versions of the HTML standard.

However, Latin 9 will not correctly encode Cyrillic, the writing system used for Russian and other Eastern European countries. A variety of 8-bit encoding schemes for Cyrillic are popular, including KOI8-R, CP1251 and CP866; ISO-8859-5 is the encoding for Cyrillic in the ISO 8859 collection.

For some languages, such as Chinese, Japanese and Korean[8], 256 numbers are simply not enough. Most of these encoding schemes use 16 bits (two bytes) to provide 65,536 different code points. To vastly simplify, the most common Chinese encodings are Big5 (often used in Taiwan and Hong Kong, for example) and GB, or Guobiao (often used on the Chinese mainland). Popular Japanese encodings include Shift JIS, EUC-JP and ISO-2022-JP. Popular Korean encodings are ISO-2022-KR and EUC-KR.

The promise of Unicode

With so many different encodings, it is difficult to choose which ones to support. The Unicode Consortium has set out to create a universal encoding—the Unicode Standard—for all characters. Their motto is "Unicode provides a unique number for every character, no matter what the platform, no matter what the program, no matter what the language." The perhaps surprising thing is this: they seem to be succeeding in creating a standard that is widely used[9]. Unicode-aware applications are able to interchange text in a consistent way, and know that a given code point corresponds to a character and vice-versa. Given our example project, which involves software running on Web servers, Web clients, email transport and email clients as well as databases, this is a very good thing. The Unicode Standard is supported by a wide variety of companies; it defines the basic text/character model for XML; and is directly supported by Java and the current generation of Microsoft operating systems and development tools.

Not limited to 128 or 256 or 65,536 different code points, the Unicode Standard asserts that every distinct character have its own code point, without regard to the specific

---

[8] Often called, collectively, the CJK languages.

[9] The International Standards Organization has also supported the Unicode Standard by working with the Unicode Consortium to create ISO standards that directly reflect the Unicode Standard.

storage layout. Version 3.2 of the standard, for example, defined 95,221 distinct characters.

In addition, the Unicode Standard defines three encoding schemes for the Unicode encoding. The schemes trade off between simplicity of access (most or all code points are represented by the same number of bits) and compactness (more common code points are represented by fewer bits than less common code points). The three standards are:

UTF-32. In UTF-32, each character's code point is represented by 32 bits. Having each character occupy a fixed width means access to a character in a Unicode string can occur quickly, at the expense of four bytes per character (meaning, that for 'typical' 8-bit ASCII strings, strings take up four times as much space). Although I mention UTF-32 first, it isn't commonly deployed. More common are UTF-8 and UTF-16.

UTF-8. Characters are represented by a varying number of bits in UTF-8—either 8, 16, 24 or 32 bits. Conveniently for applications and programmers, the ASCII characters are represented by 8 bits, and the same 8 bits as in 8-bit ASCII. So, a string such as "International Council of Cheese Producers" takes up the same amount of space in 8-bit ASCII as UTF-8[10]. By clever use of escaping, all other Unicode characters such as 'é,' '麼' and 'あ' are represented in UTF-8. UTF-8 is popular for web applications, because it potentially minimizes the amount of text that must be transported.

UTF-16. In UTF-16, characters are encoded using either 16 bits or 32 bits. Most commonly used characters (up to code point 65,535) are encoded using 16 bits; others are encoded using 32 bits. UTF-16 is the encoding used by modern Microsoft products and the more recent versions of the Microsoft operating systems[11]. Most characters fit within 16 bits; the so-called "supplemental characters" (those requiring 32 bits) are additional CJK common characters, for "private use," or for representing ancient or little-used scripts, as well as some mathematical and musical symbols. Thus, most of the time, 16

---

[10] Just to be clear: the Unicode Standard is a character standard, not a string standard, so issues such as how to determine the end of a string is not addressed in the Unicode Standard.

[11] The Unicode Standard 1.1 only allowed 2^{16} possible characters, which then meant all characters were encoded using 16 bits. At that point, another standard, UCS-2, used the same encoding and encoding scheme as UTF-16.

```
<html><head>
<META http-equiv="Content-Type" content="text/html;charset=UTF-8">
<title>国际乳酪厂商理事会</title></head>
<body>
<h1>国际乳酪厂商理事会</h1>
...
</body></html>
```

Figure 1. HTML for Chinese version of the International Cheese Poll

bits is enough. UTF-16 comes in two flavors, UTF-16-BE and UTF-16-LE, depending on which byte is more significant, the first ("big-endian") or second ("little-endian"). Currently, the default encoding for all XML documents is UTF-8, unless the first two bytes represent a Unicode "byte order mark," in which case it defaults to UTF-16[12].

How programming language and development tools support texts support the Unicode standard varies. As the Unicode Standard changes, and the languages and tools need to change as well. For example, Java, version 1.4.3 claims to use the Unicode Standard, version 3.0, but only allows code points to 2^{32}-1 (Sun, 2003a). Java 1.5, the beta for which has just become available, follows the Unicode Standard, version 4.0, and supports the full range of Unicode code points[13]. Various versions of the Microsoft Windows operating system have differing support for Unicode; Windows 2000, Windows NT, and the Microsoft .NET all use the Unicode UTF-16 encoding scheme[14].

Static Web pages

At present, the latest versions of most Internet browsers directly support the Unicode UTF-8. It makes the most sense—especially given the multilingual nature of our application—to create these pages using the UTF-8 encoding. In our scenario, our Java

[12] In hexadecimal, the byte order mark, or BOM, is xFEFF. If the first two bytes are xFE, xFF , then the default encoding is UTF-16BE; if the first two bytes are xFF, xFE, the default encoding is UTF-16LE; otherwise, if defaults to UTF-8.

[13] In Java 1.5, the Java `char` type remains limited to 16 bits. The Java `int` type is used to represent code points 2^{32} or greater (Sun, 2003b).

[14] *Developing International Software, Second Edition* by the Microsoft internationalization team (using the *nom de plume* "Dr. International") is very useful for understanding internationalization using Microsoft products (International, 2003).

```
<html><head>
<META http-equiv="Content-Type" content="text/html;charset=UTF-8">
<title>&#x56FD;&#x9645;&#x4E73;&#x916A;&#x5382;&#x5546;&#x7406;&#x4E8B;&#x4F1A;</title></head>
<body>
<h1>&#x56FD;&#x9645;&#x4E73;&#x916A;&#x5382;&#x5546;&#x7406;&#x4E8B;&#x4F1A;</h1>
...
</body></html>
```

Figure 2. Alternative HTML for Chinese version of the International Cheese Poll

program is to create static pages for each language containing the polls listed in Table 1. First, though, we should consider how to create these web pages manually, so we don't have to consider the programming and database issues involved.

Figure 1 shows a skeleton of the Chinese version of *International Polling*'s Cheese poll. The especially important line is:

```
<META http-equiv="Content-Type" content="text/html;charset=UTF-8">
```

This line, which should be placed as soon as possible in the `head` element, informs the world that the text is encoded in the UTF-8 encoding scheme[15].

Figure 2 shows an alternative version of the Chinese poll. Here, the Unicode Chinese characters are replaced with HTML entities of the form **&#x*n*;**, where *n* is the hexadecimal[16] representation, as text digits, of each character's code point. The disadvantages of this method are, first, it's very difficult for a human to read the HTML source page, and second, instead of the maximum of 4 bytes required by UTF-8, each character here requires 8 bytes (and thus, a page with a lot of Chinese it in will require at

[15] There's a bit of cleverness here. According to the HTTP 1.1 standard, the assumed encoding scheme is ISO-8895-1, the Western Latin 1 encoding, but recall that the first 127 code points and bit representations in UTF-8 are the same as the those of ISO-8895-1 (that is, the ASCII part). The `Meta` element must use readable characters from that set, so the page can be thus inform its readers that it is in UTF-8 even when the default encoding is not UTF-8.

[16] The Chinese title could also have been represented as decimal digits (*i.e.*, `&#22269;&#38469;` etc.; but that tends to be even longer.

least twice as much space to store and at least twice as much time to send, receive and process)[17].

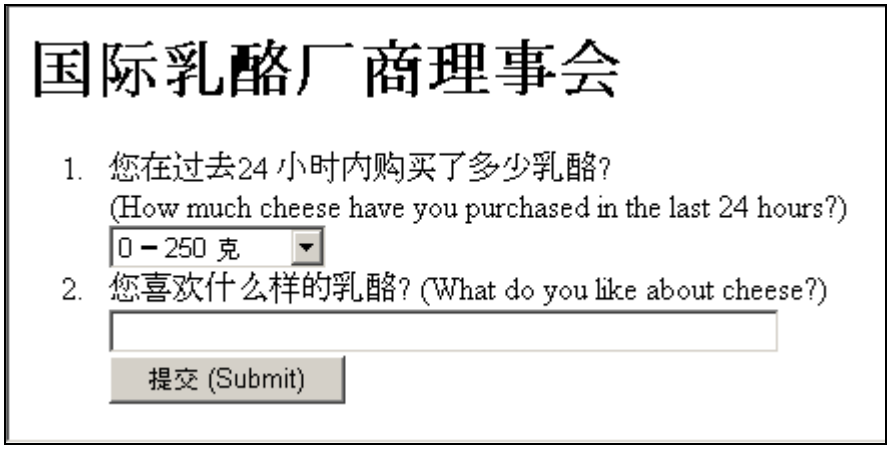

Figure 3. International Cheese Poll in Chinese

Web forms

Figure 3 shows the displayed version of the Chinese poll (with some English text for context). Because this is a web form that will receive text (the answer box to the *What do you like about cheese?* question is a `text` input), the program receiving the results must understand the encoding used. We can, in fact, control the encoding accepted by the browser by using the `accept-charset` attribute on the `form` element. The `accept-charset` attribute takes a comma-delimited list of encodings. Thus, using

```
<form accept-charset= "UTF-8" method="PUT">…</form>
```

will keep the text being sent in the UTF-8 encoding.

Web forms can use either `GET` or `PUT` methods. Using PUT and an `enctype` of `multipart/form-data` will keep the text in the encoding scheme(s) defined in the `accept-charset` attribute. The complete HTML code for the Chinese poll looks like Figure 4. The `GET` method can also be used. However, the string is encoded in yet another standard, `application/x-www-form-urlencoded`, the standard for text in URLs (Berners-Lee, Masinter and McCahill, 1994). This is fine for ASCII text, but

[17] I used Microsoft Notepad to create the Chinese poll of Figure 1. You need a Unicode/UTF-8 aware application to be able to save text in UTF-8, of course.

```
<FORM accept-charset="UTF-8" action=deleted
        method="POST"  enctype="multipart/form-data">
<ol>
<li>您在过去24 小时内购买了多少乳酪? <br/>(How much cheese have you purchased in the
last 24 hours?)<br/>
<SELECT NAME="Amt">
<OPTION SELECTED VALUE="1">0 — 250 克
<OPTION VALUE="2">250-500 克
<OPTION VALUE="3">500-750 克
<OPTION VALUE="4">750克 — 1公斤
<OPTION VALUE="5">多于1 公斤
</SELECT></li>
<li>您喜欢什么样的乳酪? (What do you like about cheese?)<br/>
<INPUT TYPE="text" MAXLENGTH="50" SIZE="50" NAME="Qualities"><br>
<INPUT TYPE="submit" value="提交 (Submit)">
</li>
</ol>
</FORM>
```

Figure 4. HTML Code for Chinese Form

more problematic for non-ASCII text: each non-ASCII code point is converted into ASCII text; for example, 国 (which requires 3 bytes in UTF-8) is represented as `%E5%9B%BD` (9 bytes). Not only is this 3 times as long and hard for humans to read, but it may exceed the limit of a server's ability to handle long URLs.

Databases

Database support for Unicode is improving, but often a little difficult to understand. In our example, *International Polling* uses the PostgreSQL database system. PostgreSQL[18] allows per-database setting of the encoding scheme, supporting UTF-8, the ISO-8859 schemes, and many others (PostgreSQL 2003).

The interface between a database system and a programming language must also support Unicode or other encoding schemes for all of this to work properly. Later versions of JDBC (Java Data Base Connection) should support Unicode directly, although specific JDBC drivers should be tested for compliance.

[18] This is from PostgreSQL version 7.4.

Email

Electronic mail was one of the first Internet applications developed; and as such, it has a good bit of legacy technology of which one must be wary. In particular, the original mail transport programs assumed that text would be transmitted in 7-bit ASCII form. Subsequent electronic mail standards are beginning to overcome these limits, while attempting to allow legacy email systems to continue to work properly[19].

There is a 7-bit encoding scheme for Unicode specifically designed for email, called UTF-7 (Goldsmith and Davis, 1997). Recent email clients from Microsoft and Netscape support this encoding[20]. Unfortunately (for our *International Polling* system), Java currently does not support UTF-7[21]. UTF-8 is also supported by recent email clients, and this is probably the best encoding scheme to use, although there might be some problems in transmission. Alternatively, code for converting UTF-8 to UTF-7 is available on the web[22].

Email uses the same MIME `content-type` specification system as HTML documents, and Goldsmith and Davis (1997) give the following example of the text portion of a message containing the Unicode sequence "A<NOT IDENTICAL TO><ALPHA>."

```
Content-Type: text/plain; charset=utf-7

A+ImIDkQ.
```

Alternatively, this could be encoded as:

```
Content-Type: text/plain; charset=utf-8

A≠A.
```

[19] Consider, for example, this quotation from Freed and Borenstein (1996): "Several of the mechanisms described in this set of documents may seem somewhat strange or even baroque at first reading. It is important to note that compatibility with existing standards *and* robustness across existing practice were two of the highest priorities of the working group that developed this set of documents. In particular, compatibility was always favored over elegance."

[20] Specifically tested were Microsoft Outlook 2000 SR-1 and Netscape 7.0.

[21] Neither Java 2, version 1.4 nor Java 2, version 1.5 support UTF-7.

[22] For example, "Java Glossary: UTF" at http://mindprod.com/jgloss/utf.html.

all of which is in UTF-8, and where the second "A" is the Greek Alpha.

Recommendations

Here are some recommendations when building multilingual systems using the Web, email and databases:

> Try to keep all text is a readable format; this will make debugging internationalization problems easier[23]. This may require upgrading to more recent versions of programming languages, editors and design tools,[24] databases, etc.
>
> Consider using the Unicode UTF-8 encoding scheme for all or most of your application. This is a good compromise (at least in the Western world) between size and compatibility.
>
> Remember that each module that processes your text must be multilingual aware.
>
> Remember that different versions of standards and software support different versions of the Unicode standard.
>
> Remember that your users will need browser and font support to see your multilingual applications.
>
> Test email applications to ensure that multilingual email is supported.

Summary

Building multilingual applications that integrate the Web, email and databases can be a daunting task. Fortunately, strong emerging support for the Unicode Standard is making this easier than it has been in the past.

---

[23] That is, it's better to see "国际乳酪厂商理事会" than it is to see "国际乳酪厂商理事会".

[24] Dreamweaver MX 2004, for example, now supports Unicode.

Acknowledgements

I'm grateful to my colleagues who provided translations for the *International Polling*'s Cheese Poll: Ilia Goldfarb (Russian), Daniel Lemire (French) and Yuhong Yan (Chinese).

Bibliography

Berners-Lee, Timothy, L. Masinter and M. McCahill, *Uniform Resource Locators (URL) (RFC 1738)*. ftp://ftp.rfc-editor.org/in-notes/rfc1738.txt, December, 1994. Assessed February, 2004.

Freed, Ned and Nathaniel Borenstein, *Multipurpose Internet Mail Extensions (MIME) part one: Format of Internet message bodies (RFC 2045)*. ftp://ftp.rfc-editor.org/in-notes/rfc2045.txt, November, 1996. Assessed February, 2004.

Goldsmith, David and Mark Davis, *UTF-7 A Mail-Safe Transformation Format of Unicode (RFC 2252)*. ftp://ftp.rfc-editor.org/in-notes/rfc2152.txt, May, 1997. Assessed February, 2004.

International, Dr. *Developing International Software, Second Edition*, Microsoft Press, 2003.

The PostgreSQL Global Development Group. *Character Set Support, PostgreSQL 7.4 Documentation*, http://www.postgresql.org/docs/7.4/interactive/multibyte.html, 2003. Accessed February, 2004.

Sun Microsystems, Inc. *Java™ 2 Platform Standard Edition v1.4.2*. http://java.sun.com/j2se/1.4.2/docs/api/, 2003. Accessed February, 2004.

Sun Microsystems, Inc. *Java™ 2 Platform Standard Edition v1.5.0*. http://java.sun.com/j2se/1.5.0/docs/api/, 2003. Accessed February, 2004.

Number	Char	Number	Char	Number	Char
32	Space	64	@	96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	'	71	G	103	g
40	(	72	H	104	h
41	)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	s
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	v
55	7	87	W	119	w
56	8	88	X	120	x
57	9	89	Y	121	y
58	:	90	Z	122	z
59	;	91	[	123	{
60	<	92	\	124	|
61	=	93	]	125	}
62	>	94	^	126	~
63	?	95	_	127	

ASCII Codes from 32 to 127.

Appendix 1: ASCII Character Table

The table below shows the ASCII encoding for ASCII code points 32 to127. These are also the Unicode code points as well. Values under 31 are “control characters,” the most commonly used being 0 (null character), 9 (horizontal tab), 10 (line feed), and 13 (carriage return). All numbers are in decimal.