Introduction to Unicode

Presented by developerWorks, your source for great tutorials

ibm.com/developerWorks

Table of Contents

If you're viewing this document online, you can click any of the topics below to link directly to that section.

Section 1. Tutorial basics

Is this tutorial right for you?

This tutorial is for anyone who wants to understand the basics of Unicode-based multilingual Web page development. It is an introduction to how multilingual characters, the Unicode encoding standard, XML, and Perl CGI scripts can be integrated to produce a truly multilingual Web page. The tutorial explains the concepts behind this process and lays the groundwork for future development.

Navigation

Navigating through the tutorial is easy:

* Use the Next and Previous buttons to move forward and backward.
* Use the Menu button to return to the tutorial menu.
* If you'd like to tell us what you think, use the Feedback button.
* If you need help with the tutorial, use the Help button.

What is required: Hardware and software needed

No additional hardware is needed for this tutorial; however, in order to view some of the files online, Unicode fonts must be loaded. See Resources for information on Unicode fonts if you don't already have one loaded. (Note: Many Unicode fonts are still incomplete or in development, so one particular font may not contain every character; however, for the purposes of this tutorial, the Unicode font downloaded by the user will ideally contain all or most of the language characters used in our examples. (If yours does not, try downloading an additional Unicode font.) It may be necessary to change the encoding in the browser to Unicode (UTF-8). In Internet Explorer, do this by going to *View* > *Encoding* > *Unicode (UTF-8)* .

Section 2. Introduction

Introduction to the tutorial

Multilingual computing is key to the future worldwide growth of the Internet, and Unicode and XML are significant building blocks upon which it will be built.This tutorial shows you some of the basic principles involved in constructing a multilingual Web page in Unicode.The tutorial focuses on a sample survey question in three languages, which is merely an example of what will be possible in future multilingual computing.

Multilingual software needed to produce similar Web pages

To produce your own multilingual Web page in Unicode, you'll need some type of multilingual word processing software. One of the very best resources for assessing current Unicode-based software is *Alan Wood's Unicode Resources* (see Resources). To be most efficient, one should use multilingual software that: (1) uses a Unicode font that includes all of the languages desired; (2) saves the multiple languages for the multilingual Web page as an HTML file with UTF-8 (Unicode) encoding; and (3) allows for easy editing of the HTML file containing the Unicode hexadecimal equivalents of all of the different foreign language characters. The hexadecimal equivalents can be entered by hand if one has the time to look up all the necessary Unicode references, but this is a very cumbersome procedure. The software used to produce the examples in this tutorial supports more than a hundred languages.

Terms used

ASCII - Acronym for the "American Standard Code for Information Interchange." As an assignment standard for English-based characters, ASCII is structured on the Latin character set, upon which English is based. CGI - Protocol that stands for "Common Gateway Interface." The CGI protocol in this case is the mechanism and definition for how browsers communicate with servers. Perl is a programming language; a "Perl CGI script" is a script written in the Perl language which adheres to and uses the CGI protocol. Unicode - Universal encoding standard for (eventually) all the characters and symbols used in all the languages of the world, past and present. It has merged with and is compatible with international standard ISO/IEC 10646. Unicode is a registered trademark of Unicode, Inc., and is in development by the Unicode Consortium. The latest version is 3.0 (released in early 2000). While currently still in development at about 50,000 characters, the Unicode standard will eventually be capable of displaying more than a million characters, enough to cover all the living and ancient languages in the world and all required scientific and mathematical and other types of symbols. A Unicode font is usually a subset of this larger standard and contains "glyphs." Glyphs are representations of characters within a particular font. For example, the English letter "D" can also be printed as *D*. These are different glyphs for the same character.XML - eXtensible Markup Language (XML) is merely a markup language like HTML which uses some specific conventions but then allows for nearly limitless creativity and flexibility in the use of self-identified tags. This ability is a tremendous organizing principle which is largely lacking in HTML. XML allows for the creation of sub-languages and complexity but also easy retrieval and manipulation of the data thus organized.

About the author

Jim Melnick is president of Internet Interactive Services, which does multilingual Web design and consulting. His e-mail address is: info@PortableExpert.com.You can also visit his Web site at *http://www.PortableExpert.com* .

Section 3. Unicode-based multilingual form development

Translation

The first step in constructing a Unicode-based multilingual Web page is fairly self-evident: The material must be translated into the desired target languages by persons knowledgeable in those languages. At some point in the future, automatic translation or global translation (formerly known as "machine translation" or "MT") may be sophisticated enough to do a large part of that job, but it is not quite ready at this time. Additionally, although great strides have been made in this area in recent years, it is hard to imagine a time when human review of automatically translated text will not be necessary.

Localization

Localization is the next important concept in understanding multilingual computing. Web localization can be defind as simply the act of making a Web site linguistically and culturally appropriate to a local audience. An "accurate" translation may not be enough -- a translated text must be "localized" for the target audience viewing the Web page. We could use Spanish as an example: A Web page might be accurately translated into Spanish, but then the question could be, Which Spanish? Peruvian Spanish, for example, is not the same as Mexican Spanish. Therefore, Peruvians reading a Mexican Spanish Web site might be able to understand almost all of it, but certain nuances or turns of phrase might be unfamiliar to them. For the most part, multilingual sites are not yet sophisticated enough or targeted enough to deal with such differentiations, but that will certainly change in the future. In other words, localization will become more and more significant as it helps direct the growth and acceptance of the Internet in ever more broad and diverse cultural settings.

Multiple languages on the same Web page

For the purposes of this tutorial, however, localization is not an issue: The English used is clearly American (not British), the Russian text used is very straightforward, and although Chinese has many different spoken dialects, any Chinese reader could understand and respond to the survey question as presented. Our main goal here is to simply construct the initial building blocks of a multilingual Web site, where it is possible to display multiple languages *on the same Web page together*. Once the key concepts involved in getting a Unicode-based multilingual Web page up and running are understood, localization and more advanced aspects of Web design can be addressed by the developer.

Expert review

Expert review is rather self-explanatory. After a Web page has been translated, globalized for consistency and content, and localized for individual target language impact, it should undergo expert review. Does the site hang together overall? Is there a consistent message for content and tone between the languages? This examination for consistency between languages might also be viewed as the process of "internationalizing" the site. These are all issues that could fall under the category of expert review.

Page markup

After translation, localization and expert review, we can proceed to working out the Unicode equivalents and the actual page markup. Besides the three languages -- English, Russian and Chinese -- used for the survey question in this tutorial, we have also added random characters in Japanese, Hebrew, Hindi, and Tibetan to demonstrate the amazing variety of Unicode-based characters in a multilingual site.

Sample multilingual survey question

We begin to construct our Unicode-based multilingual Web page example with a sample survey question: "Do you want to buy a new computer? Yes___ No___" translated (except for English, of course!) into our two other target languages. The result is displayed in Figure 1.

<English>
Do you want to buy a new computer? Yes___ No___

<Russian>
Вы хотите купить новый компьютер? Да___ Нет ___

<Chinese>
你 要 买 新 电脑？ 是 ___ 否 ___

"Unicodization"

Once this procedure is complete, we need to transfer these language texts into their Unicode equivalents. We could call this step "Unicodization" (I don't know if this term has been coined yet, but if not, it needs to be.) It is not necessary, of course, to translate the English characters of our example into their Unicode equivalents, since they would be displayed properly in any case because of ASCII. However, we do so anyway in order to demonstrate how the process works overall (as well as for consistency).

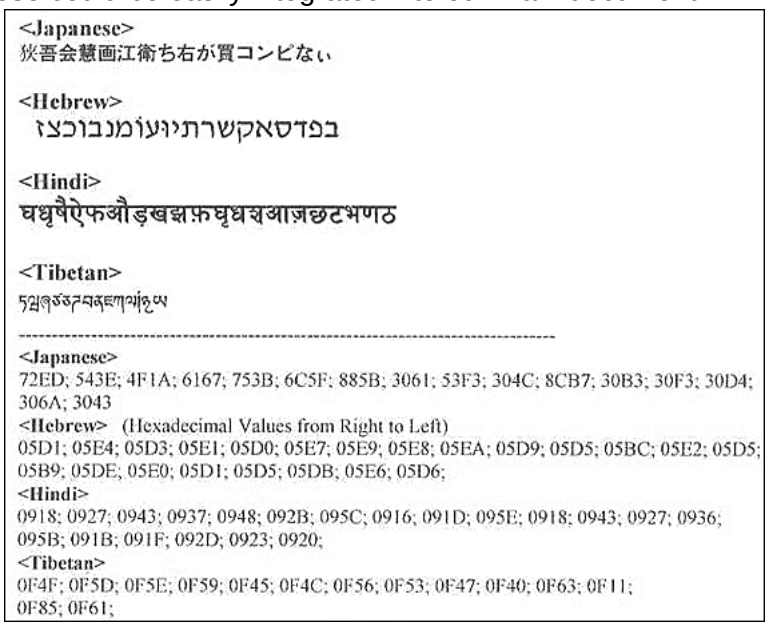
Transferring Unicode characters into their hexadecimal equivalents

Although Unicode does work with decimal numbers, hexadecimal numbers are the standard. The Unicode characters are transferred into their hexadecimal equivalents. The characters (with the underlying hexadecimal equivalents) are then placed by the software or by hand into whatever markup document is being prepared. The key point is that, whether the user or developer sees them or not, the hexadecimal equivalents are the foundation of the process, and can then be manipulated as needed for various other programming purposes. See Figure 2 for the result of our particular example.

<English>	
0044; 006F;	Do
0079; 006F; 0075;	you
0077; 0061; 006E; 0074;	want
0074; 006F;	to
0062; 0075; 0079;	buy
0061;	a
006E; 0065; 0077;	new
0063; 006F; 006D; 0070; 0075; 0074; 0065; 0072; 003F;	computer
003F;	?
0059; 0065; 0073;	Yes
004E; 006F;	No
<Russian>	
0412; 044B;	Вы
0445; 043E; 0442; 0438; 0442; 0435;	хотите
043A; 0443; 043F; 0438; 0442; 044C;	купить
043D; 043E; 0432; 044B; 0439;	новый
043A; 043E; 043C; 043F; 044C; 044E; 0442; 0435; 0440;	компьютер
003F;	?
0414; 0430	Да
041D; 0435; 0442;	Нет
<Chinese>	
4F60;	你
8981;	要
4E70;	买
65B0;	新
7535; 8111;	电 脑
003F;	?
662F;	是
5426;	否

Random characters in other languages

In the previous example we used three languages. However, Unicode allows us to use a large number of languages -- at least in short sentences or segments -- on the same Web page. See Figure 3 for some sample random characters in other languages. We chose Japanese, Hebrew, Hindi, and Tibetan to demonstrate some of the amazing versatility of Unicode. The latter portion of Figure 3 shows the hexadecimal equivalents of those characters. If our survey question had been translated into all of these languages, these could be easily integrated into our main document.



Adding Unicode hexadecimal numbers to the page markup

The next step is transferring these Unicode hexadecimal numbers into markup language for the Web page which will be built around them. To do this, we add the symbols &#x to the front of the number with a semi-colon (;) placed at the end. For example, the Chinese characters for the word "computer" are designated as follows in hexadecimal form: `&#x7535; &#x8111;`.

Using XML tags

XML tags provide a basis for organization and for building complexity into multilingual documents. For the purpose of this tutorial, we simply use a single survey question as the basis of our Unicode-based multilingual document. However, as multilingual e-commerce develops, documents can easily become extremely complex. XML can provide an excellent mechanism for managing this process (although, as we shall see in a moment, it is still not all worked out). Construct XML tags to "grow" with the document, which may ultimately involve various multilingual interfaces or applications.

No multilingual standard for XML

There is, however, a problem: At present, there is not a consistent multilingual XML standard. As Yves Savourel points out in an article in the October/November 2000 edition of *MultiLingual Computing and Technology* magazine, "a standard markup method is needed for working with multilingual documents" ("XML Technologies and the Localization Process," #35, Volume 11, Issue 7, p. 62). Savourel's comment could eventually emerge as a major understatement! Just as Unicode itself is bringing standardization to the characters of the world's languages and symbols, a standard XML language for multilingual documents will become crucial to the smooth development of multilingual e-commerce.

Form development: More complex scripts and XML tags needed

For the purposes of our simple survey, the lack of a multilingual XML standard is not a problem. A more complex multilingual survey might be devised that would have numerous questions and would be sorted and tabulated by XML according to language, type of response, region of the world, or other factors. The user would first find his language, then work his way down through a series of questions, with answers and responses being sent back and forth to a cgi bin file. For now, we will merely use a simple Perl CGI script. In a more complex multilingual Web page, numerous layers of scripts and responses might be utilized. Defining how those scripts are used and interact -- and using XML in that process -- is at the heart of building effective multilingual Web sites using Unicode. That is a more advanced topic that builds on what we have presented here.

Display of HTML

Now we're ready to view our basic survey question. For those who have not yet loaded a Unicode font, Figure 4 displays survey3.htm similarly to how it appears online with Unicode loaded into the browser:
Figure 4

If you have a Unicode font already loaded, you can view this online directly at:
PortableExpert.com/survey3.htm

Section 4. Going interactive: Adding the Perl CGI script

The markup

The markup for survey3.htm is structured as follows:
<HTML> <H1>Sample Survey Question in Three Languages</H1>

<FORM ACTION="http://www.PortableExpert.com/cgi-bin/survey3.cgi"
METHOD=POST>

<English>   D o   y o u
#x00A0;w a n t   t o
#x00A0; b u y   a
#x006E; e w   c o m p
#x0075; t e r ?   <INPUT TYPE="RADIO"
NAME="English" VALUE="Y e s">Yes
#x00A0; <INPUT TYPE="RADIO" NAME="English" VALUE="No">No
          <INPUT TYPE="submit"
NAME="EnglishSubmit" VALUE="Submit"></H2> </English>

<Russian> <H3> В ы   х о
#x0442; и т е   к у
#x043F; и т ь   н о
#x0432; ы й   к о м п
#x044C; ю т е р ?
INPUT TYPE="RADIO" NAME="Russian"
#x0430;   <INPUT TYPE="RADIO" NAME="Russian"
VALUE="Нет">Нет
#x00A0;   <INPUT TYPE="submit" NAME="RussianSubmit"
#x041F;уск"> </H3> </Russian>

<Chinese> <H2> 你 要 买 新
#x8111;?   <INPUT TYPE="RADIO" NAME="Chinese"
VALUE="是">是   <INPUT TYPE="RADIO"
NAME="Chinese" VALUE="否">否
INPUT TYPE="submit" NAME="ChineseSubmit" VALUE="提
> </Chinese> </FORM> </HTML>

Figure 5

Survey questions in hexadecimal format and notional XML structure

Figure 5 shows the survey questions in their hexadecimal form in an HTML document. It also demonstrates how a notional XML tag structure could be set up for future expansion that would allow for sorting by language, by response or by other factors set up by the developer.

Placing a file in the cgi bin

You are now ready to go interactive! We have created a Perl CGI script* which activates a file in the cgi bin subdirectory when any of the three "submit" buttons on survey3.htm in either English, Russian or Chinese is pressed. The file in the cgi-bin (survey3.cgi) is set up in such a way that the three languages follow each other consecutively with no other formatting. This is for demonstration purposes only. The script returns the following words in all three languages: "desktop computer, laptop computer, and palmtop computer" (again, you must have Unicode-loaded fonts to view this properly). *NOTE: For those unfamiliar with Perl CGI scripting, please note the following:

1. The lines in a Perl CGI script are not numbered.
2. The scripts are placed in the cgi bin subdirectory within the overall directory in use. In this case, the script is executing on a UNIX server. The HTML file pulls up the CGI file to be executed when one of the submit buttons is pressed.
3. When the Perl CGI script is first created, it must be transferred in ASCII format (not binary!) to the cgi bin file of the directory.
4. Appropriate commands and permissions must be made: a chmod (UNIX) command to the file to establish read, write, and execute permissions, as desired; and chmod 755 (for example, chmod 755 survey3.cgi).
5. For more information, please consult materials on Perl and CGI, such as Elizabeth Castro's *Perl and CGI for the World Wide Web* (Peachpit Press, 1999).

Form method and input type

In our example, we use the form method of "POST," while the input type is a radio button for the "Yes" or "No" options in the various languages. Radio buttons permit only one choice in a set. However, in our example, every choice returns the same answer.

More hexadecimals

The values for the words representing "Yes" and "No" have been translated into the Unicode hexadecimal numbers for those words in each target language, surrounded by quotation marks (note also that ` ` is the hexadecimal equivalent for the *space* character).

---

survey3.cgi

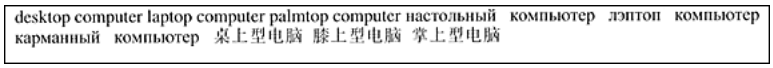
The Perl CGI script file "survey3.cgi" is placed in a cgi-bin subdirectory. It is located at: www.PortableExpert.com/cgi-bin/survey3.cgi.The Perl CGI script for "survey3.cgi" is similar to the following:

```
#!/usr/local/bin/perl print "Content-type:
text/html\n\n"; # English section* print"desktop computer laptop
computer palmtop computer"; # Russian section
#x0430;&#x0441;&#x0442;&#x043E;
&#x0439;   &#x043A;&#x043E;&#x043C;
#x044C;&#x044E;&#x0442;&#x0435; &#x0440; 
&#x043F;&#x0442;&#x043E; &#x043F;
#x043E;&#x043C; &#x043F;&#x044C;&#x044E;&#x0442;&#x0435;
  &#x043A;&#x0430;&#x0440;&#x043C;&#x0430;
#x043D;&#x044B;&#x0439;   &#x043A;&#x043E;&#x043C;
&#x043F;&#x044C;&#x044E;&#x0442;&#x0435; &#x0440; "; #
Chinese section print"&#x684C;&#x4E0A;&#x578B;&#x7535;&#x8111;
#x819D;&#x4E0A;&#x578B;&#x7535;&#x8111;
&#x7535;&#x8111;";
```

**In this section of the tutorial example, we have printed out the words in English rather than transferring them also into their hexadecimal equivalents.*

CGI script execution returns a multilingual result

When the Perl CGI script is executed, it returns the following result:
Figure 6



This demonstrates that the Perl CGI script, when executed, will return Unicode hexadecimal equivalents as multilingual characters back to a browser.

Just the first step

This is just the first step in developing interactivity using Unicode-based multilingual characters. The next steps would obviously include equating the values in the "Yes" and "No" responses for each of the languages with corresponding responses from the Perl CGI script. For example, if the answer to the question in Russian is "No," the script could lead to some sort of concluding question in Russian. If the answer is "Yes," the next response in Russian (or Chinese or English) would be: "Would you like to purchase a: (1) desktop computer; (2) laptop computer; or (3) palmtop computer?" Then, based on which selection was made, the user would follow other threads in that language, leading him or her to some final result. XML tags could be used to tabulate the data and responses from each of the different languages and then format them in some master document for assessment and comparison. However, the next steps will form the basis of a follow-up article or tutorial.

Section 5. Conclusions and resources

Conclusions: What have you learned?

In this tutorial you have learned the very basic construction techniques of a Unicode-based multilingual Web page. Through the use of XML, Unicode, and programming languages such as Perl, there are great possibilities for developing broad-based multilingual interactivity, as well as for organizing data and tabulating responses in many different languages. As the Unicode Standard Version 3.0 states, Unicode "defines a consistent way of encoding multilingual text that enables the exchange of text data internationally and creates the foundation for global software" (*The Unicode Standard Version 3.0*, p. 1, 2000). We have merely scratched the surface in demonstrating how this global software might be developed, but it will certainly follow many of the principles presented in this tutorial.

Conclusions: Problem areas

There are, nevertheless, still problem areas. As Michel Rodriguez points out in "Character Encodings in XML and Perl" (see Resources), certain current applications and databases do not readily accept Unicode input. There are various techniques (using appropriate encoding tables and mapping) for the required input and output conversions, but these can be cumbersome. Additionally, it will still take some time before Unicode fonts and XML-enabled browsers are standard worldwide. Regardless of these problems, Unicode-based multilingual development will soon begin to grow rapidly, providing programmers and Web developers with a tremendous number of possibilities in software applications that are truly global.

Resources

* For background information on Unicode multilingual computing, see my previous developerWorks article, *Multilingual forms in Unicode: Developing Unicode-based global software applications* .
* To view Unicode characters for the hexadecimal numbers designated in this tutorial, a Unicode font such as Code2000, Bitstream Cyberbit, or Arial Unicode must be loaded into "Fonts" on your browser. An excellent resource is *Alan Wood's Unicode Resources*, at *Windows Fonts that Support Unicode* . You can also obtain a very fine Unicode shareware font from *James Kass* , 424 W. Commonwealth #304, Fullerton, CA 92832.
* Another portion of Alan Wood's Unicode Resources site, entitled Unicode and Multilingual Editors and Word Processors is probably the best compilation available on multilingual word-processing software with Unicode capabilities. There are various types of capabilities which come with different software packages, including Microsoft's FrontPage 2000, Unitype's Global Writer, Netscape Composer 6, SUE (Simple Unicode Editor) for the Macintosh, UnicEdit for Windows NT 4.0, Yudit, and many others. Readers are encouraged to check out the Wood site extensively and to experiment with various types of software listed there based on their specific needs.
* Find information on the new Unicode 3.0 Standard at *Unicode.org* .
* The *MultiLingual Computing & Technology* Web site contains some information; however, it does not contain most back articles. MultiLingual Computing & Technology's address is: MultiLingual Computing, Inc., 319 North First Avenue, Sandpoint, ID 83864.
* For an excellent article and examples of some of the issues involved in using Perl, XML and Unicode, see Michel Rodriguez's "Character Encodings in XML and Perl" at *XML.com* (April, 2000).

Section 6. Feedback

Your feedback

Please let us know whether this tutorial was helpful to you and how we could make it better. We'd also like to hear about other tutorial topics you'd like to see covered. Thanks!

Colophon

This tutorial was written entirely in XML, using the developerWorks Toot-O-Matic tutorial generator. The Toot-O-Matic tool is a short Java program that uses XSLT stylesheets to convert the XML source into a number of HTML pages, a zip file, JPEG heading graphics, and PDF files. Our ability to generate multiple text and binary formats from a single source file illustrates the power and flexibility of XML.