MOBISOFT
DISCOVER MOBILITY

# Understanding Character Encodings

...

Basics of Character Encodings that all Programmers should Know.

Pritam Barhate, Cofounder and CTO Mobisoft Infotech.

MOBISOFT
DISCOVER MOBILITY

# Introduction

- I am Pritam Barhate. I am cofounder and CTO of Mobisoft Infotech, which is a iPhone & iPad Application Development Company.
- Mobisoft also has expertise in creating scalable cloud based API backends for iOS and Android applications.
- For more information about various services provided by our company, please check our website.

MOBISOFT
DISCOVER MOBILITY

# Definitions

- Character

  A character is a sign or a symbol in a writing system. In computing a character can be, a letter, a digit, a punctuation or mathematical symbol or a control character (these are not visible, for example, carriage return).

- Script

  A script is a collection of letters and other written signs used to represent textual information in one or more writing systems. For example, Latin is a script which supports multiple languages like English, French and German.

# The Need for Character Sets

- Computers only understand binary data. To represents the characters as required by human languages, the concept of character sets was introduced.
- In character sets each character in a human language is represented by a number.
- In early computing English was the only language used. To represent, the characters used in English, ASCII character set was used.
- ASCII used 7 bits to represent 128 characters which included: numbers 0 to 9, lowercase letters a to z, uppercase letters A to Z, basic punctuation symbols, control codes that originated with Teletype machines, and a space.
- For example, in ASCII, character O is represented by decimal number 79. Same can be written in binary format as: 1001111 or in hexadecimal (hex) format as: 4F.

MOBISOFT
DISCOVER MOBILITY

# Evolution!

- Obviously ASCII was insufficient to represent characters in all human languages. This led to creation to multiple character sets as computing evolved and became more widespread throughout the world. However, as computing evolved, multiple companies created different computing platforms which had support for different character sets.
- Various attempts were made by different platform providers to extend the currently supported character sets while maintaining backward compatibility. This created a lot of confusion when data created by program using X encoding was to be used in a program which uses Y encoding as its default.
- Finally Unicode emerged as a common standard to support majority of written scripts by people throughout the world.

MOBISOFT
DISCOVER MOBILITY

# Some Commonly Used Character Sets

- ASCII
- Unicode
- Windows 1252
- ISO 8859

# So if I use Unicode I am OK, Right?

MOBISOFT
DISCOVER MOBILITY

# UTF-8, UTF-16 and UTF-32

- Since there are thousands of characters in Unicode, just one byte is not sufficient to represent every character.
- Hence multiple bytes are required to represent Unicode characters.
- Though 32 bits or 4 bytes are sufficient to represent every character in Unicode, most of the data is in English and Western European languages. So using uniform 32 bits everywhere is waste of a lot of space.
- Hence, UTF-8 and UTF-16 are more prevalent. These encodings use variable number of bytes to represent each character as needed.
- UTF-8 is the most prevalent encoding used on web.
- All modern Windows OSes and Java SE 5.0 onwards use UTF-16.

MOBISOFT
DISCOVER MOBILITY

# How do Encodings Affect Programming?

- When data is transferred between various computing systems, you need to carefully check in which encoding the data is coming to your system and in which encoding you are providing the data to other systems.
- An error in this generally results part or all of data being interpreted as garbage.

MOBISOFT
DISCOVER MOBILITY

# Tips to Handle Encoding Properly in your Programs - I

- When saving your web pages, which are read by browsers, make sure that you are using the correct encoding while saving your file. Most of the text editors have the encoding setting buried deep somewhere in preferences/options. Take out the time to learn how your favorite text editor / IDE handles file encodings.
- In static HTML files, as well as dynamically generated HTML always use proper charset attribute value for meta tag. Make sure that you are indeed outputting the data in the character set mentioned by you. For example:

```
<head><meta charset="UTF-8"></head>
```

MOBISOFT
DISCOVER MOBILITY

# Tips to Handle Encoding Properly in your Programs - II

- While outputting data for dynamic web pages and APIs make sure that right Content-Type header is being sent to the client with the response.

  For example: `Content-Type: text/html; charset=utf-8`

- While creating the database in MySQL or PostgreSQL, make sure that you specify the correct character encoding for your tables.
- For new programs, which use MySQL, when you create the database for ‘Default Collation’ setting choose: utf8 - utf8_unicode_ci.
- For new systems which use PostgreSQL, when you create the database choose UTF-8 as encoding and en_US.UTF-8 as collation.

MOBISOFT
DISCOVER MOBILITY

# Tips to Handle Encoding Properly in your Programs - III

- While importing data from text based files, such as CSV, make sure that you detect the encoding of the file and then read the data using that encoding. Then while saving the data in your system, make sure that you have converted to the character set that your system uses. If you are using a library to do the CSV reading and writing for you, take out the time to understand how the library handles character encodings. Choose a library only after making sure that it has a robust support for various character encodings.
- Take out time to understand how the programming language / libraries, you are using, handle character encoding while parsing files and parsing API output such as JSON / XML.

MOBISOFT
DISCOVER MOBILITY

# How to detect Character Encoding Reliably

- Unfortunately detecting character encoding of files is not an exact process because of the way it has evolved over the years on multiple computing platforms.
- The ICU - International Components for Unicode project provides APIs which can help you to detect Character Encoding of a file in a reliable way. They have bindings available for most of the popular programming languages.
- ICU also provides APIs to convert data from one character encoding to another.
- Also if your programming language has multiple libraries to detect character encoding then you can chain them together as fallback if in case the primary library fails to detect the character encoding.

MOBISOFT
DISCOVER MOBILITY

# Useful PHP Functions to Handle Character Encodings

- Most the PHP functions to read files (for example, `file_get_contents`) return data as string.
- Unfortunately in PHP strings are just byte arrays. They don't store any encoding information. You can use following functions to detect the encoding of a string.

  `mb_detect_encoding` — Detect character encoding

  `mb_convert_encoding` — Convert character encoding

- However, the implementations of these functions have been criticized online. So use them with caution.

MOBISOFT
DISCOVER MOBILITY

# How to Handle Character Encoding in Java

- There are multiple libraries to detect encodings in Java. This stackoverflow question has a good discussion about this. This answer from the question has a given a good strategy to use multiple libraries when one fails to detect the character encoding of a given file. In general, ICU4J seems to be quite reliable.
- Also in Java, you can use `InputStreamReader.getEncoding()`to detect encoding.
- Once you have a byte array of data and you have detected the character encoding for it, you can use the following String constructor to create a string with proper encoding: `public String(byte[] bytes, Charset charset)`
- To save a file with desired character encoding, you need to pass the proper character set to the OutputStreamWriter class. Here is an example of how to do it.

MOBISOFT
DISCOVER MOBILITY

# How to Handle Character Encoding in Objective-C

- Since Objective-C can easily call C functions, you can use the ICU4C to detect character encoding.
- Another option is to use the UniversalDetector library to detect character encodings. This library is a wrapper for uchardet, which is based on C++ implementation of the universal charset detection library by Mozilla.
- Once you have detected the encoding of a byte buffer, you can pass that encoding to `NSString` constructor to create the correct string with data. For example: - `(instancetype)initWithData:(NSData *)data encoding:(NSStringEncoding)encoding`
- To write a string to a file with desired encoding: - `(BOOL)writeToFile:(NSString *)path atomically:(BOOL)useAuxiliaryFile encoding:(NSStringEncoding)enc error:(NSError * _Nullable *)error`

MOBISOFT
DISCOVER MOBILITY

# How to Handle Character Encoding in C#

- To detect character encodings in C#, you can use the chardetsharp library, which is a port of Mozilla's CharDet Character Set Detector.
- Also it should be possible to create native bindings to ICU4C. You can find more information about this here.
- Once you have detected the character set of the byte array, you can use something similar to this example to obtain a correct C# string object from the data.
- To write a C# string to a file with a desired encoding, you can follow the examples provided here.

MOBISOFT
DISCOVER MOBILITY

# Thank You!