

Unicode: Os segredos da Codificação de Caracteres

Douglas V. Pasqua

douglas.pasqua@gmail.com

<http://douglaspasqua.com>



Objetivo

- Entendendo o que é e como funciona a codificação de caracteres
- Identificando e resolvendo problemas comuns de codificação de caracteres.
- Entendendo o Unicode (UTF-8, UTF-16 e UTF-32)
- Trabalhando com codificação de caracteres na Web / PHP / Banco de Dados / IDEs
- Dicas de Migração de codificação de caracteres

Tópicos

- Introdução e Teoria

- Entendendo a codificação de caracteres
- No principio tudo era ASCII
- Codificações de 8-bits
- Unicode
- Codificações baseadas em Unicode
 - UTF-8
 - UTF-16
 - UTF-32
- Por que usar UTF-8 ?

Tópicos

- Entendendo problemas comuns de codificação
- Codificação de caracteres na Web
- Codificação de caracteres e PHP
- Codificação de caracteres e Banco de Dados
- Codificação de caracteres e Editores

Tópicos

- Introdução e Teoria
 - Entendendo a codificação de caracteres
 - No principio tudo era ASCII
 - Codificações de 8-bits
 - Unicode
 - Codificações baseadas em Unicode
 - UTF-8
 - UTF-16
 - UTF-32
 - Por que usar UTF-8 ?

Codificação de caracteres

- A codificação de caracteres diz ao computador como interpretar zeros e uns em caracteres *reais*.

$$01000001 = 65 = A$$

Codificação de caracteres

- Os mais freqüentes/conhecidos são:
 - ASCII (Utiliza 7bits)
 - 8-bit encodings
 - ISO-8859-1 (Latin1)
 - ISO-8859-15 (French)
 -
 - Unicode
 - UTF-8
 - UTF-16
 - UTF-32

Tópicos

- Introdução e Teoria

- Entendendo a codificação de caracteres
- **No principio tudo era ASCII**
- Codificações de 8-bits
- Unicode
- Codificações baseadas em Unicode
 - UTF-8
 - UTF-16
 - UTF-32
- Por que usar UTF-8 ?

ASCII

- Codificação que define 128 caracteres
(0-127) ou (00000000 - 01111111)
- Baseado no alfabeto Inglês
- 1 Byte para representar todos caracteres
- 7 bits de uso. O bit + significativo é sempre zero (0)

ASCII

- O bit + significativo é sempre 0 (Ajuda validar a integridade de caracteres transmitidos)

01111111

Tabela ASCII

Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0		32	20	40	[space]	64	40	100	@	96	60	140	`
1	1	1		33	21	41	!	65	41	101	A	97	61	141	a
2	2	2		34	22	42	"	66	42	102	B	98	62	142	b
3	3	3		35	23	43	#	67	43	103	C	99	63	143	c
4	4	4		36	24	44	\$	68	44	104	D	100	64	144	d
5	5	5		37	25	45	%	69	45	105	E	101	65	145	e
6	6	6		38	26	46	&	70	46	106	F	102	66	146	f
7	7	7		39	27	47	'	71	47	107	G	103	67	147	g
8	8	10		40	28	50	(72	48	110	H	104	68	150	h
9	9	11		41	29	51)	73	49	111	I	105	69	151	i
10	A	12		42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	13		43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14		44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15		45	2D	55	-	77	4D	115	M	109	6D	155	m
14	E	16		46	2E	56	.	78	4E	116	N	110	6E	156	n
15	F	17		47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20		48	30	60	0	80	50	120	P	112	70	160	p
17	11	21		49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22		50	32	62	2	82	52	122	R	114	72	162	r
19	13	23		51	33	63	3	83	53	123	S	115	73	163	s
20	14	24		52	34	64	4	84	54	124	T	116	74	164	t
21	15	25		53	35	65	5	85	55	125	U	117	75	165	u
22	16	26		54	36	66	6	86	56	126	V	118	76	166	v
23	17	27		55	37	67	7	87	57	127	W	119	77	167	w
24	18	30		56	38	70	8	88	58	130	X	120	78	170	x
25	19	31		57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32		58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33		59	3B	73	;	91	5B	133	[123	7B	173	{
28	1C	34		60	3C	74	<	92	5C	134	\	124	7C	174	
29	1D	35		61	3D	75	=	93	5D	135]	125	7D	175	}
30	1E	36		62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37		63	3F	77	?	95	5F	137	_	127	7F	177	

ASCII - Considerações

- Define alguns caracteres não imprimíveis.
- Atende muito bem a língua inglesa.
- Não é muito útil para nosso idioma, principalmente porque não define caracteres com acentuação.

Tópicos

- Introdução e Teoria

- Entendendo a codificação de caracteres
- No principio tudo era ASCII
- **Codificações de 8-bits**
- Unicode
- Codificações baseadas em Unicode
 - UTF-8
 - UTF-16
 - UTF-32
- Por que usar UTF-8 ?

Codificações de 8-bits

- Codificação de caracteres de 8 bits são uma extensão do ASCII
- Utiliza 1 byte assim como o ASCII
- Utiliza todos os 8-bits
- Dobro da capacidade em relação ao ASCII (256 caracteres)

ISO-8859-1

- Codificação 8 bits mais comum encontrada nos sites da internet.
- Possui todos os caracteres usados nos países da Europa Ocidental (Incluindo Língua Portuguesa)
- Conhecida informalmente como [Latin-1](#)
- Mantém compatibilidade com tabela ASCII (Primeiros 128 caracteres)

ISO-8859-1

- Compõe o alfabeto Latino
- Base da codificação Windows-1252 (Western European)
- Range de Caracteres:
 - 0 à 127 (ASCII)
 - 128 à 159 (Não utilizado)
 - 160 à 255

ISO-8859-1

Char	Code	Char	Code	Char	Code	Char	Code	Char	Code	Char	Code	Char	Code	Char	Code
	160	ı	161	ç	162	£	163	¤	164	¥	165		166	§	167
¨	168	©	169	ª	170	«	171	¬	172		173	®	174	¯	175
°	176	±	177	²	178	³	179	´	180	µ	181	¶	182	·	183
,	184	¹	185	º	186	»	187	¼	188	½	189	¾	190	¿	191
À	192	-	193	Â	194	Ã	195	Ä	196	Å	197	Æ	198	Ç	199
È	200	É	201	Ê	202	Ë	203	Ì	204	Í	205	Î	206	Ï	207
Ð	208	Ñ	209	Ò	210	Ó	211	Ô	212	Õ	213	Ö	214	×	215
Ø	216	Ù	217	Ú	218	Û	219	Ü	220	Ý	221	Þ	222	ß	223
à	224	á	225	â	226	ã	227	ä	228	å	229	æ	230	ç	231
è	232	é	233	ê	234	ë	235	ì	236	í	237	î	238	ï	239
ð	240	ñ	241	ò	242	ó	243	ô	244	õ	245	ö	246	÷	247
ø	248	ù	249	ú	250	û	251	ü	252	ý	253	þ	254	ÿ	255

Séries ISO-8859

ISO-8859-1 - Latin 1

Western Europe and Americas: Afrikaans, Basque, Catalan, Danish, Dutch, English, Faeroese, Finnish, French, Galician, German, Icelandic, Irish, Italian, Norwegian, Portuguese, Spanish and Swedish.

ISO-8859-2 Latin 2

Latin-written Slavic and Central European languages: Czech, German, Hungarian, Polish, Romanian, Croatian, Slovak, Slovene.

ISO-8859-3 - Latin 3

Esperanto, Galician, Maltese, and Turkish.

ISO-8859-4 - Latin 4

Scandinavia/Baltic (mostly covered by 8859-1 also): Estonian, Latvian, and Lithuanian. It is an incomplete predecessor of Latin 6.

Tópicos

- Introdução e Teoria

- Entendendo a codificação de caracteres
- No principio tudo era ASCII
- Codificações de 8-bits
- **Unicode**
- Codificações baseadas em Unicode
 - UTF-8
 - UTF-16
 - UTF-32
- Por que usar UTF-8 ?

Unicode - Características

- Implementa o padrão Unicode através:
 - UTF-8
 - UTF-16
 - UTF-32
- 1.114.112 de *code points*
- Pode utilizar mais que 8-bits (até 4bytes)
- Suporta praticamente todos idiomas do mundo

Unicode

- Atualmente é a codificação de caracteres mais comum na Web
- Possui mais de 100mil símbolos mapeados
 - Símbolos matemáticos
 - Formas geométricas
 - Ideogramas japoneses
 - Verificar + símbolos <http://www.unicodetables.com/>

Tópicos

- Introdução e Teoria

- Entendendo a codificação de caracteres
- No principio tudo era ASCII
- Codificações de 8-bits
- Unicode
- Codificações baseadas em Unicode
 - UTF-8
 - UTF-16
 - UTF-32
- Por que usar UTF-8 ?

UTF-8

- É um tipo de codificação Unicode de byte variável
- Pode utilizar de 1 até 4 bytes
 - **1 Byte:** compátivel com tabela ASCII (128)
 - **2 Bytes:**
 - suficiente para caracteres Latinos
 - alfabetos Grego, Hebraico, Sirio, Armênico, etc.
 - **3 Bytes:** Plano Multilingual Básico.(Chinês, Japonês, Coreano)
 - **4 Bytes:** Alguns outros caracteres. (símbolos)

UTF-8 usando 1 byte (Exemplo)

7 bits efetivos / Decimal 76 / Símbolo: *L*

01001100



01001100

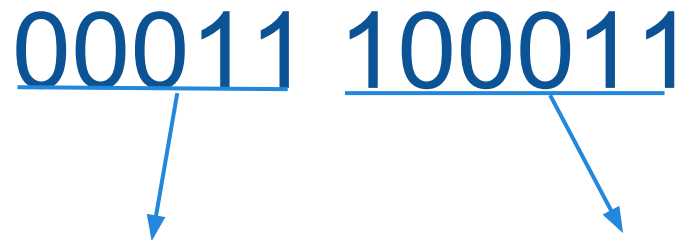
U+0000 até U+007F / 0 até **127**

128 Caracteres

UTF-8 usando 2 bytes (Exemplo)

11 bits efetivos / Decimal: 227 / Símbolo: ã

00011 100011



11000011 - **10**100011

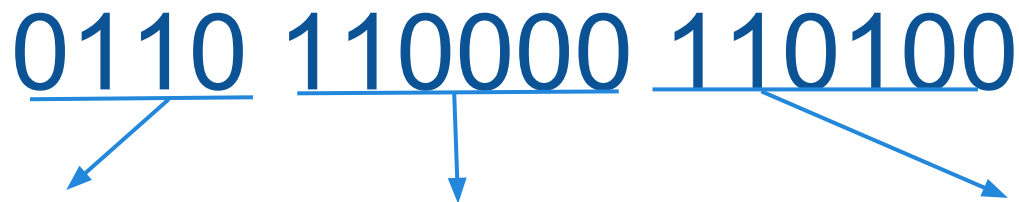
U+0080 até U+07FF / 128 até 2047

1919 Caracteres

UTF-8 usando 3 bytes

16 bits efetivos / Decimal 27700 / Símbolo: 水

0110 110000 110100



11100110 - 10110000 - 10110100

U+0800 até U+FFFF / 2.048 até 65.535

63.488 Caracteres

UTF-8 usando 4 bytes

21 bits efetivos

11110xxx - 10xxxxxx - 10xxxxxx - 10xxxxxx

65536 até 1.114.111

U+10000 até U+10FFFF

1.048.576 Caracteres

Tópicos

- Introdução e Teoria

- Entendendo a codificação de caracteres
- No principio tudo era ASCII
- Codificações de 8-bits
- Unicode
- Codificações baseadas em Unicode
 - UTF-8
 - **UTF-16**
 - UTF-32
- Por que usar UTF-8 ?

UTF-16

- Codificação Unicode de tamanho de byte variável
- Utiliza 2 ou 4 bytes
- Eficiente para textos escritos em idiomas utilizados em países Asiáticos

UTF-16

- 2 bytes: (Range)

U+0000 até U+FFFF / 0 - 65.535

- 4 bytes: (Range)

U+10000 to U+10FFFF / 65.536 até 1.114.111

Tópicos

- Introdução e Teoria

- Entendendo a codificação de caracteres
- No principio tudo era ASCII
- Codificações de 8-bits
- Unicode
- Codificações baseadas em Unicode
 - UTF-8
 - UTF-16
 - **UTF-32**
- Por que usar UTF-8 ?

UTF-32

- Codificação Unicode de tamanho de bytes fixo.
- Utiliza sempre 4 bytes
- Símbolos são facilmente indexados
- Utiliza espaço de forma ineficiente
- Raramente utilizado

UTF-32

- 4 Bytes: (Range)

U+0000 até U+10FFFF / 0 até 1.114.111

UTF-32 Exemplo

Character: Z

Decimal: 90

ASCII / ISO-8859-1 / UTF-8: 01011010

UTF-32: 00000000 00000000 00000000 01011010

Tópicos

- Introdução e Teoria

- Entendendo a codificação de caracteres
- No principio tudo era ASCII
- Codificações de 8-bits
- Unicode
- Codificações baseadas em Unicode
 - UTF-8
 - UTF-16
 - UTF-32
- Por que usar UTF-8 ?

Por que usar UTF-8?

- **Internacionalização**
 - Português, Inglês, Japonês, Chinês, etc.
- **Suporte**
 - Navegadores, IDEs, editores, database, etc.
- Compátivel com tabela **ASCII**
- **Compacto** - trabalhar com o mínimo de **1 byte**
- Engloba **todos** caracteres **Unicode**
- Codificação de caracteres mais popular na **Web**

Tópicos

- Entendendo problemas comuns de codificação
 - Exemplo simples de problema de codificação
 - Melhor prática para evitar problemas de codificação

Problemas comuns de codificação

Caracter ã, Decimal 227, salvo em UTF-8:

11000011 - 10100011

O que aconteceria se imprimir em ISO-8859-1?

Problemas comuns de codificação

```
1 <?php
2 header('Content-type: text/html; charset=ISO-8859-1');
3 echo "ã";
4
```

Ã £



11000011



10100011

Tópicos

- Entendendo problemas comuns de codificação
 - Exemplo simples de problema de codificação
 - Melhor prática para evitar problemas de codificação

Problemas comuns de codificação

- Procurar manter tipo de codificação única:
 - Editor / IDE
 - Web (html, http headers)
 - Aplicação (PHP)
 - Banco de Dados
 - Qualquer outro meio externo onde houver Leitura / Escrita de dados
 - memcached
 - APIs
 - RSS feeds

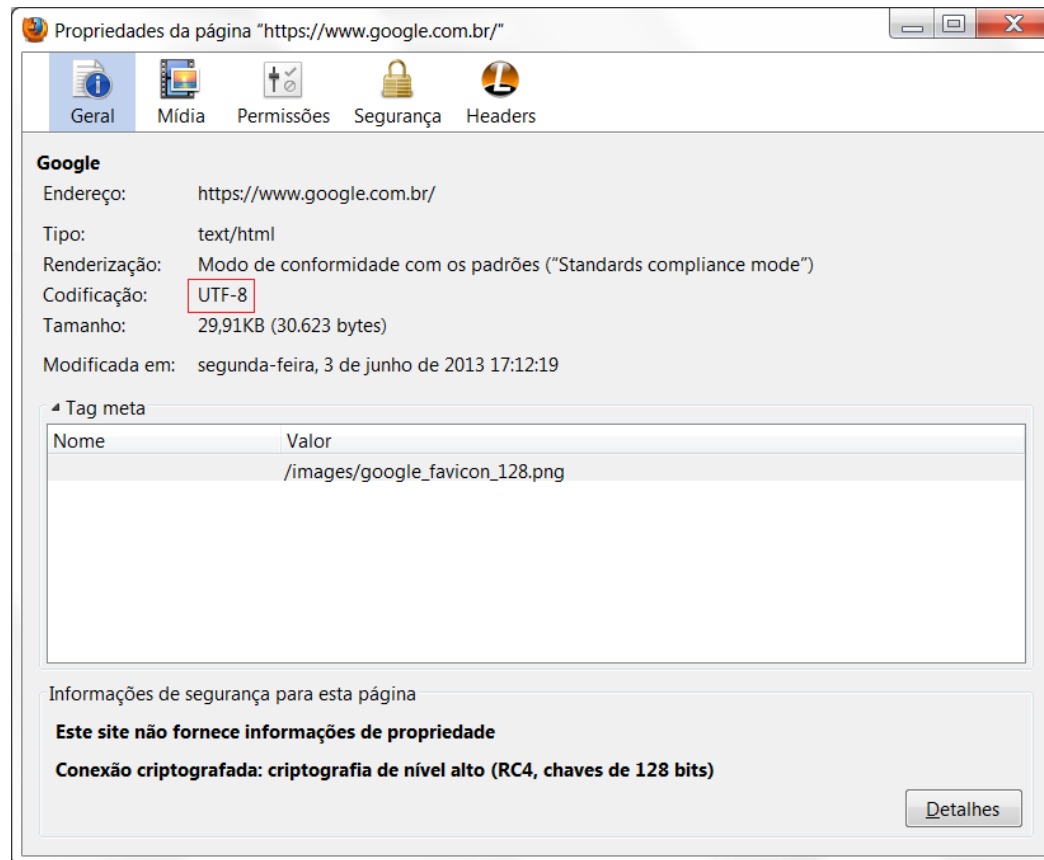
Tópicos

- Codificação de caracteres na Web
 - Descobrindo a codificação através do Navegador
 - Especificando a codificação no webserver com PHP
 - Especificando a codificação nas páginas HTML

Codificação de caracteres na Web

- Firefox:

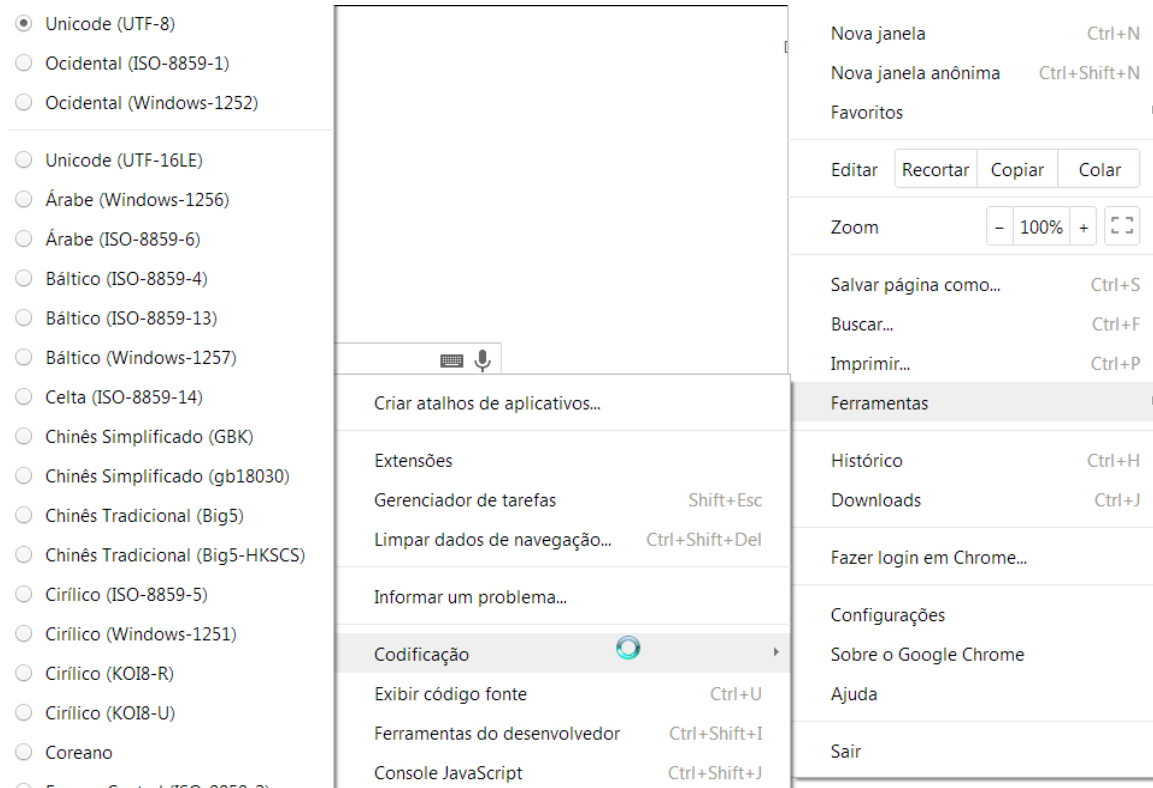
(Ferramentas -> Propriedades da Página)



Codificação de caracteres na Web

- Chrome:

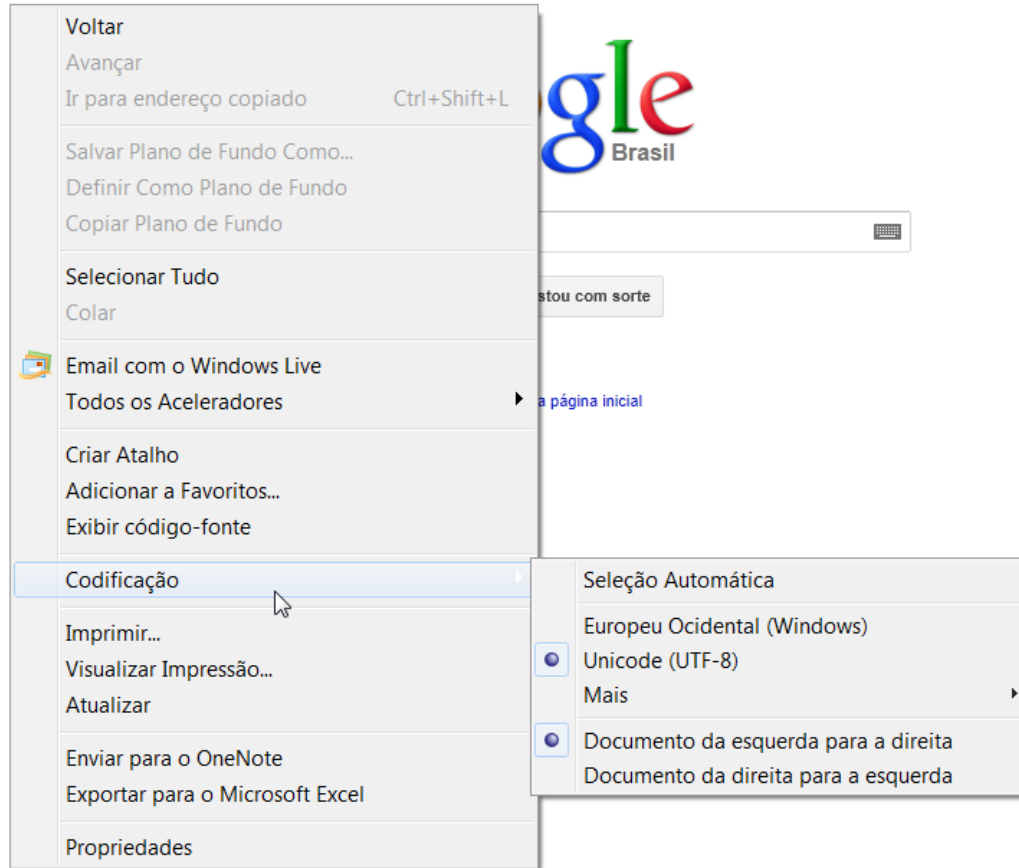
(Menu -> Ferramentas -> Codificação)



Codificação de caracteres na Web

- IE:

(Exibir -> Codificação)



Tópicos

- Codificação de caracteres na Web
 - Descobrindo a codificação através do Navegador
 - **Especificando a codificação no webserver com PHP**
 - Especificando a codificação nas páginas HTML

Codificação de caracteres na Web

- Especificando no webserver com PHP

- função header:

```
1 <?php
2 header('Content-type: text/html; charset=UTF-8');
3
```

- php.ini:

```
; PHP's default character set is set to empty.
; http://php.net/default-charset
default_charset = "UTF-8"
```

Codificação de caracteres na Web

- Especificando no webserver com PHP

- função ini_set:

```
1 <?php
2 ini_set('default_charset', 'UTF-8');
```

- usando .htaccess:

```
php_value default_charset "UTF-8"
```


Tópicos

- Codificação de caracteres na Web
 - Descobrindo a codificação através do Navegador
 - Especificando a codificação no webserver com PHP
 - Especificando a codificação nas páginas HTML

Codificação de caracteres na Web

- Especificando no HTML (meta tag)

```
<meta http-equiv='Content-Type' content='text/html; charset=utf-8'>
```

- HTML5:

```
<meta charset='utf-8'>
```

- XHTML/XML:

```
<?xml version="1.0" encoding="UTF-8"?>
```

Codificação de caracteres na Web

- Especificando no HTML (meta tag)
 - A codificação especificada no header do HTTP (php ou apache) tem precedência em relação à meta tag
 - Procurar especificar a mesma codificação na meta tag em relação ao especificado no header (PHP ou Apache)

Codificação de caracteres na Web

- Especificando no HTML (meta tag)
 - Boas razões para especificar a meta tag no HTML:
 - Caso o usuário fizer o download da página local.
 - Por algum motivo, a codificação não ter sido enviado pelo webserver (php ou apache)

Tópicos

- Codificação de caracteres e PHP
 - **PHP e Unicode**
 - utf8_encode / utf8_decode
 - Usando extensão mbstring
 - Conversão de codificação
 - Expressões Regulares
 - PHP 6 e Unicode

PHP e Unicode

- Strings nativas em PHP não tem suporte a Unicode
 - Strings são tratadas como dados *binários*
- É possível contornar a situação através:
 - `utf8_encode()`
 - `utf8_decode()`
 - Extensão Multibyte String (`mbstring*`)

PHP e Unicode

Exemplo: Arquivo salvo em UTF-8

```
<?php
echo strlen("123456789ã"); // 11

echo substr("123456789ã", 9, 1); // 💎

echo substr("123456789ã", 9, 2); // ã

$var = "123456789ã";

echo $var[9] . $var[10]; // ã

echo $var[9]; // 💎
```

PHP e Unicode

Tratamento usando utf8_encode/utf8_decode

```
<?php  
  
$var = utf8_decode("123456789ã");  
  
echo strlen($var); // 10  
  
$char = substr($var, 9, 1);  
echo utf8_encode($char); // ã  
  
echo utf8_encode($var[9]); // ã
```


PHP e Unicode

Tratamento usando a extensão mbstring

```
<?php  
  
echo mb_strlen("123456789ã", "UTF-8"); // 10  
  
echo mb_substr("123456789ã", 9, 1, "UTF-8"); // ã
```

PHP e Unicode

Tratamento usando a extensão mbstring

```
<?php  
mb_internal_encoding("UTF-8");  
  
echo mb_strlen("123456789ã"); // 10  
  
echo mb_substr("123456789ã", 9, 1); // ã
```

Tópicos

- Codificação de caracteres e PHP
 - PHP e Unicode
 - **utf8_encode / utf8_decode**
 - Usando extensão mbstring
 - Conversão de codificação
 - Expressões Regulares
 - PHP 6 e Unicode

PHP - utf8_encode/utf8_decode

- utf8_encode
 - Converte uma string em ISO-8859-1 para UTF-8

```
<?php
```

```
$str_iso = str_from_databaso_iso_8859_1();
```

```
echo utf8_encode($str_iso);
```

PHP - utf8_encode/utf8_decode

- utf8_decode
 - Converte uma string em UTF-8 para ISO-8859-1

```
<?php
```

```
$str_iso = utf8_decode($str_utf8);
```

```
echo strlen($str_iso);
```

Tópicos

- Codificação de caracteres e PHP
 - PHP e Unicode
 - utf8_encode / utf8_decode
 - **Usando extensão mbstring**
 - Conversão de codificação
 - Expressões Regulares
 - PHP 6 e Unicode

PHP - Extensão mbstring

- Provê suporte multibyte para strings no PHP
- Já vem instalada por padrão na maioria das distribuições
- Ao compilar à partir do código fonte, habilitar:
 - `--enable-mbstring`

PHP - Extensão mbstring

- mb_strlen
- mb_split
- mb_strpos
- mb_strtr
- mb_strtolower
- mb_strtoupper
- mb_substr
- mb_convert_encoding

http://www.php.net/manual/pt_BR/ref.mbstring.php

Tópicos

- Codificação de caracteres e PHP
 - PHP e Unicode
 - utf8_encode / utf8_decode
 - Usando extensão mbstring
 - **Conversão de codificação**
 - Expressões Regulares
 - PHP 6 e Unicode

PHP - Conversão de codificação

- Usando extensão mbstring:

```
// converte Windows-1252 para UTF-8
$csv_utf8 = mb_convert_encoding($csv, 'UTF-8', 'Windows-1252');

// lista de codificações suportadas pelo mbstring
print_r(mb_list_encodings());
```

PHP - Conversão de codificação

- Usando extensão iconv:

```
$text = "Euro symbol '€'.";

echo 'Original : ', $text, PHP_EOL;
// Original : Euro symbol '€'.

echo 'TRANSLIT : ', iconv("UTF-8", "ISO-8859-1//TRANSLIT", $text), PHP_EOL;
// TRANSLIT : Euro symbol 'EUR'.

echo 'IGNORE    : ', iconv("UTF-8", "ISO-8859-1//IGNORE", $text), PHP_EOL;
// IGNORE     : Euro symbol ''.

echo 'Plain      : ', iconv("UTF-8", "ISO-8859-1", $text), PHP_EOL;
// Plain       : Euro symbol '
```

PHP - Conversão de codificação

- Obter lista de codificações suportadas pelo iconv:

```
$ iconv -l
```

Tópicos

- Codificação de caracteres e PHP
 - PHP e Unicode
 - utf8_encode / utf8_decode
 - Usando extensão mbstring
 - Conversão de codificação
 - **Expressões Regulares**
 - PHP 6 e Unicode

PHP - Regex e UTF-8

- funções `ereg*` não suportam UTF-8
 - deprecated
- funções `preg_*` suportam UTF-8
 - adicionando a flag `/u`

PHP - Regex e UTF-8

- Fazer o match baseado em uma sequencia Unicode
 - Utilizar o formato `\x{NNN}`
 - `U+NNN` - Número hexadecimal variável

Símbolo EURO = € -> `\x{20AC}`

```
$text = "Euro €";  
echo preg_match('/\x{20AC}/u', $text) , PHP_EOL;
```

PHP - Regex e UTF-8

- Fazer o match baseado em uma propriedade
 - `\p{L}` -> Representa qualquer Letra Unicode
 - `\p{N}` -> Representa qualquer número Unicode
 - `\p{P}` -> Representa qualquer pontuação Unicode
 - `\p{Greek}` -> Caracteres gregos

<http://php.net/manual/en/regexp.reference.unicode.php>

<http://www.regular-expressions.info/unicode.html>

Tópicos

- Codificação de caracteres e PHP
 - PHP e Unicode
 - utf8_encode / utf8_decode
 - Usando extensão mbstring
 - Conversão de codificação
 - Expressões Regulares
 - **PHP 6 e Unicode**

PHP 6 e Unicode

- Discussão sobre o suporte à Unicode desde 2001
- Inicialmente seria adotado UTF-16 internamente
 - Abortado por Ramus Ledorf em 2010
 - Reiniciado o desenvolvimento de suporte Unicode
- 2 Tipos de String
 - unicode
 - binary

Tópicos

- Codificação de caracteres e Banco de Dados
 - Criando bases UTF-8
 - Criando Tabelas UTF-8
 - Conexão com bases usando UTF-8
 - Migração de codificação de banco de dados

Banco de Dados - criando base UTF-8

- MySQL:

```
CREATE DATABASE mydatabase CHARACTER SET utf8  
COLLATE utf8_Unicode_ci;
```

- PostgreSQL:

```
CREATE DATABASE mydatabase WITH ENCODING 'UTF8';
```

Tópicos

- Codificação de caracteres e Banco de Dados
 - Criando bases UTF-8
 - **Criando Tabelas UTF-8**
 - Conexão com bases usando UTF-8
 - Migração de codificação de banco de dados

Banco de Dados - tabelas UTF-8

- MySQL:

```
CREATE TABLE usuario (  
    id INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,  
    login VARCHAR(250) NOT NULL,  
    PRIMARY KEY(id)  
) TYPE=InnoDB CHARACTER SET utf8 COLLATE utf8_general_ci;
```

Tópicos

- Codificação de caracteres e Banco de Dados
 - Criando bases UTF-8
 - Criando Tabelas UTF-8
 - **Conexão com bases usando UTF-8**
 - Migração de codificação de banco de dados

Banco de Dados - acessando base UTF-8

- MySQL:

```
<?php
```

```
// usando a extensão mysql  
mysql_set_charset("utf8", $link);
```

```
// usando a extensão mysqli  
mysqli_set_charset($link, "utf8");
```

```
// usando a extensão mysqli OOP  
$mysqli = new mysqli("localhost", "my_user", "my_password", "testdb");  
$mysqli->set_charset("utf8");
```

```
// usando PDO, PHP Version < PHP 5.3.6  
$dbh = new PDO("mysql:dbname=testdb;host=127.0.0.1", $user, $password);  
$dbh->exec("set names utf8");
```

```
// usando PDO, PHP Version >= PHP 5.3.6  
$dbh = new PDO("mysql:dbname=testdb;host=127.0.0.1;charset=utf8", $user, $password);
```


Banco de Dados - acessando base UTF-8

- PostgreSQL:

```
<?php
```

```
// extensão pgsql
```

```
pg_set_client_encoding($link, "UNICODE");
```

```
// PDO
```

```
$pdo = new PDO('pgsql:host=127.0.0.1;dbname=testdb', $user, $password);
```

```
$pdo->query("SET NAMES 'UNICODE'");
```

Tópicos

- Codificação de caracteres e Banco de Dados
 - Criando bases UTF-8
 - Criando Tabelas UTF-8
 - Conexão com bases usando UTF-8
 - Migração de codificação de banco de dados

Banco de Dados - Migrando para utf8

- Mysql:

--- Database

```
ALTER DATABASE mydatabase CHARACTER SET utf8 COLLATE  
utf8_Unicode_ci;
```

-- Tabelas

```
ALTER TABLE mytable CONVERT TO CHARACTER SET utf8 COLLATE  
utf8_Unicode_ci;
```

Banco de Dados - Migrando para utf8

- PostgreSQL (Migração método 1)
 - Fazer o dump do banco :
`$ pg_dump -f testdb.sql testdb`
 - Realizar a conversão de codificação :
`$ iconv testdb.sql -f ISO-8859-1 -t UTF-8 -o testdb_utf8.sql`

Banco de Dados - Migrando para utf8

- PostgreSQL (Migração método 2)
 - Gerar o dump com a codificação desejada:
`$ pg_dump -E utf-8 -f testdb_utf8.sql testdb`

Banco de Dados - Migrando para utf8

- PostgreSQL

- Criar a base em UTF-8:

- > CREATE DATABASE testdb_utf8 with encoding 'UTF8';

- Restaurar o banco de dados convertido:

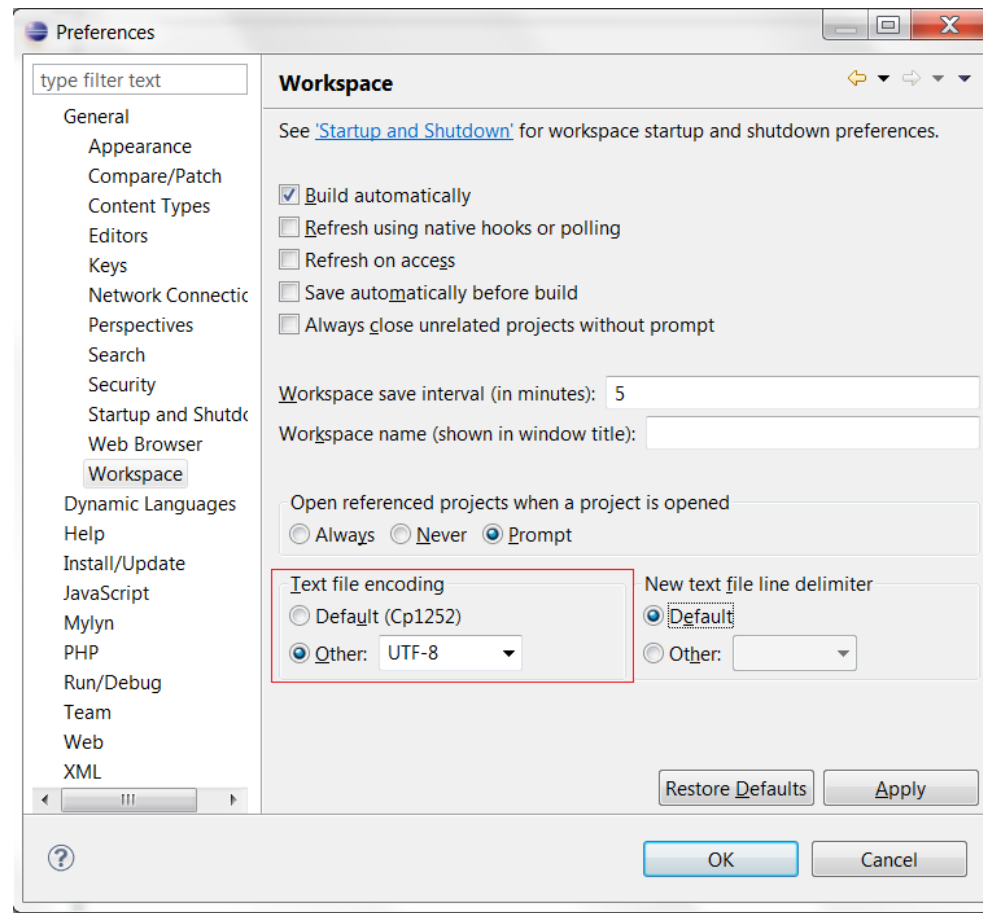
- \$ psql -d testdb_utf8 -f testdb_utf8.sql

Tópicos

- Codificação de caracteres e Editores
 - Eclipse
 - Netbeans
 - vim
 - Notepad++

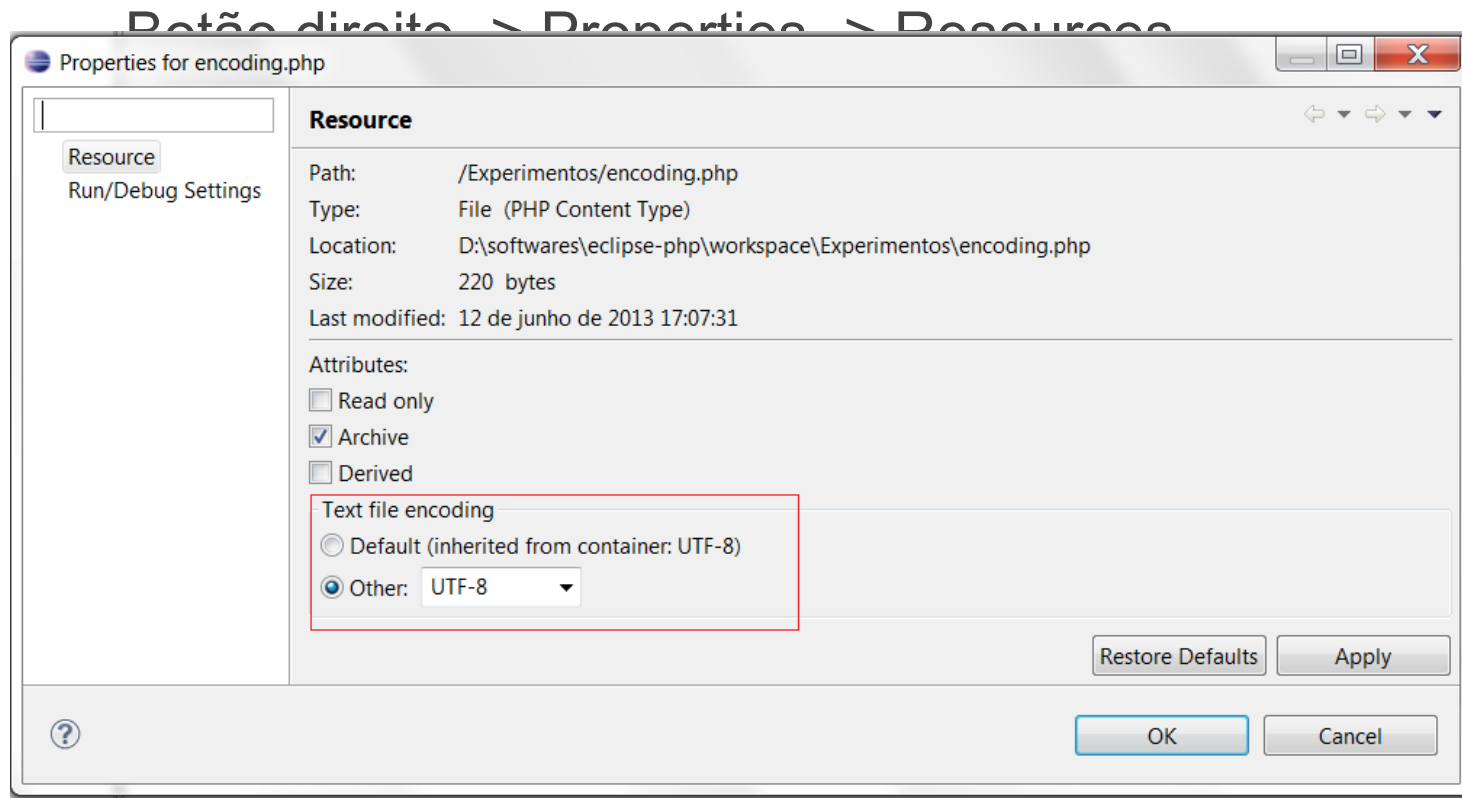
Eclipse - UTF-8

- Window->Preferences->General->Workspace



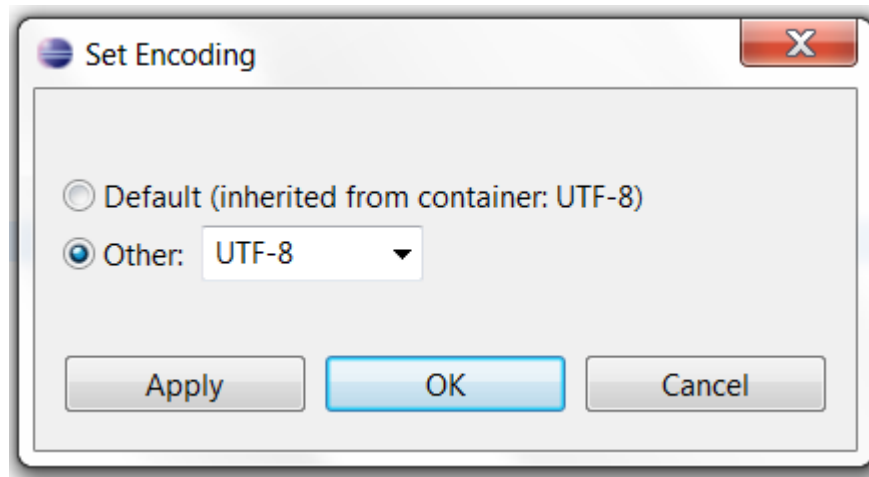
Eclipse - UTF-8

- Alterar codificação específica (arquivo, diretório, projeto)



Eclipse - UTF-8

- Aba aberta e selecionada
Edit -> Set Encoding

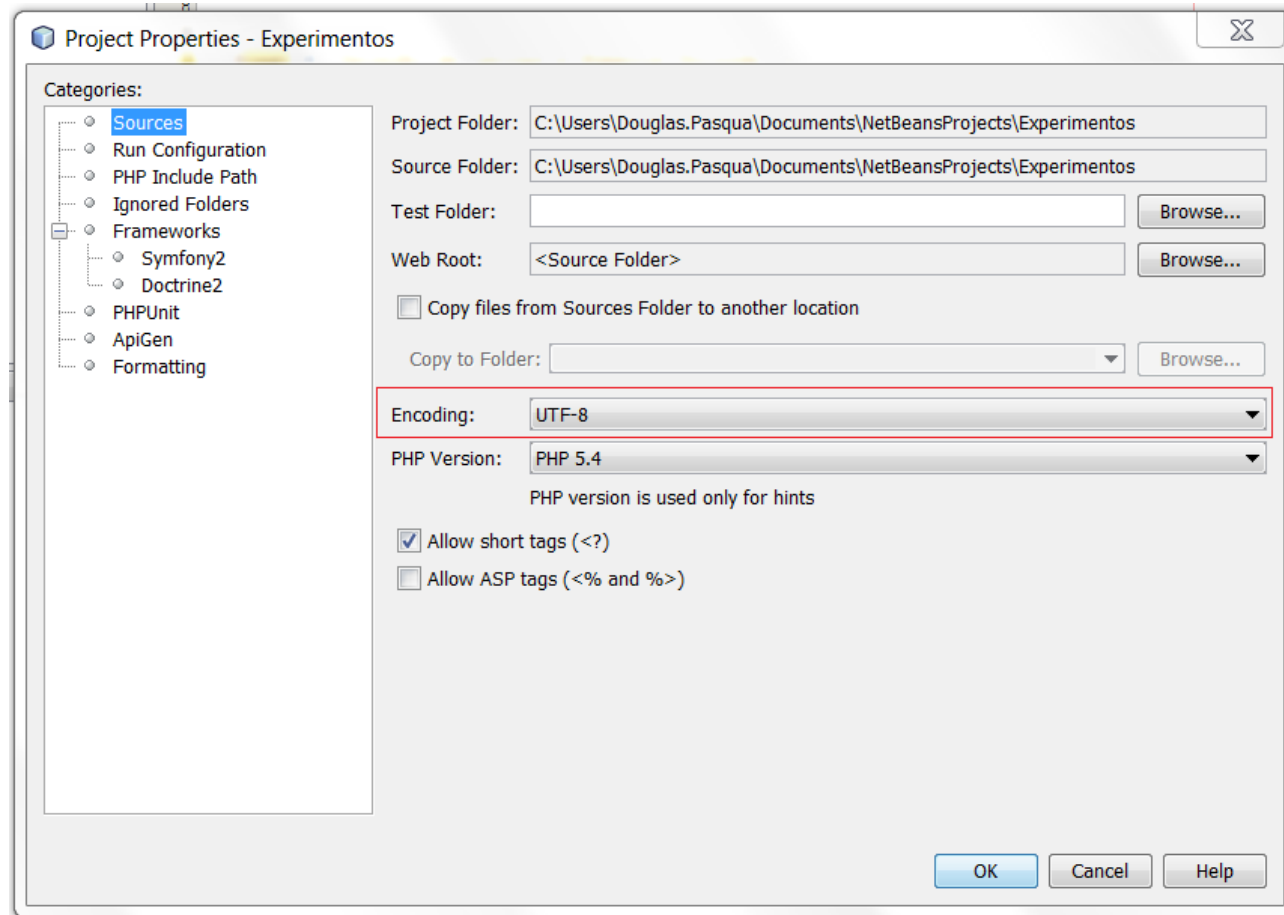


Tópicos

- Codificação de caracteres e Editores
 - Eclipse
 - **Netbeans**
 - vim
 - Notepad++

Netbeans 7 - UTF-8

- Botão direito Projeto -> Properties -> Sources

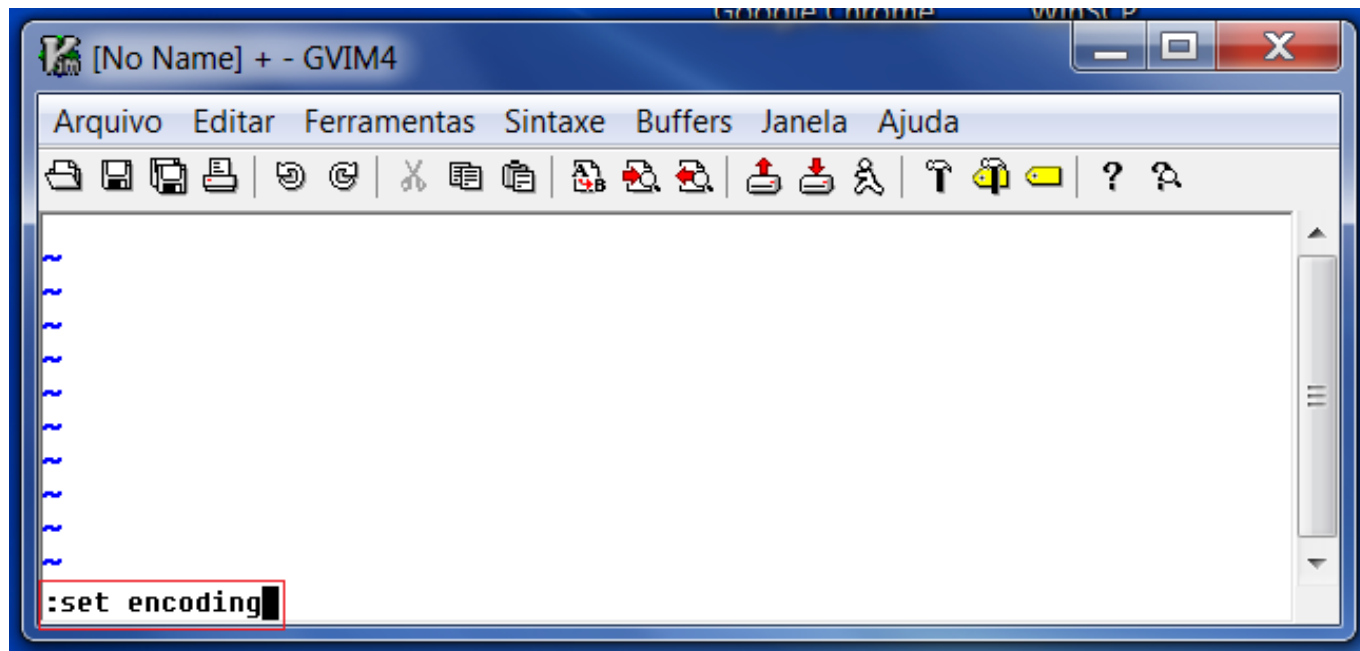


Tópicos

- Codificação de caracteres e Editores
 - Eclipse
 - Netbeans
 - **vim**
 - Notepad++

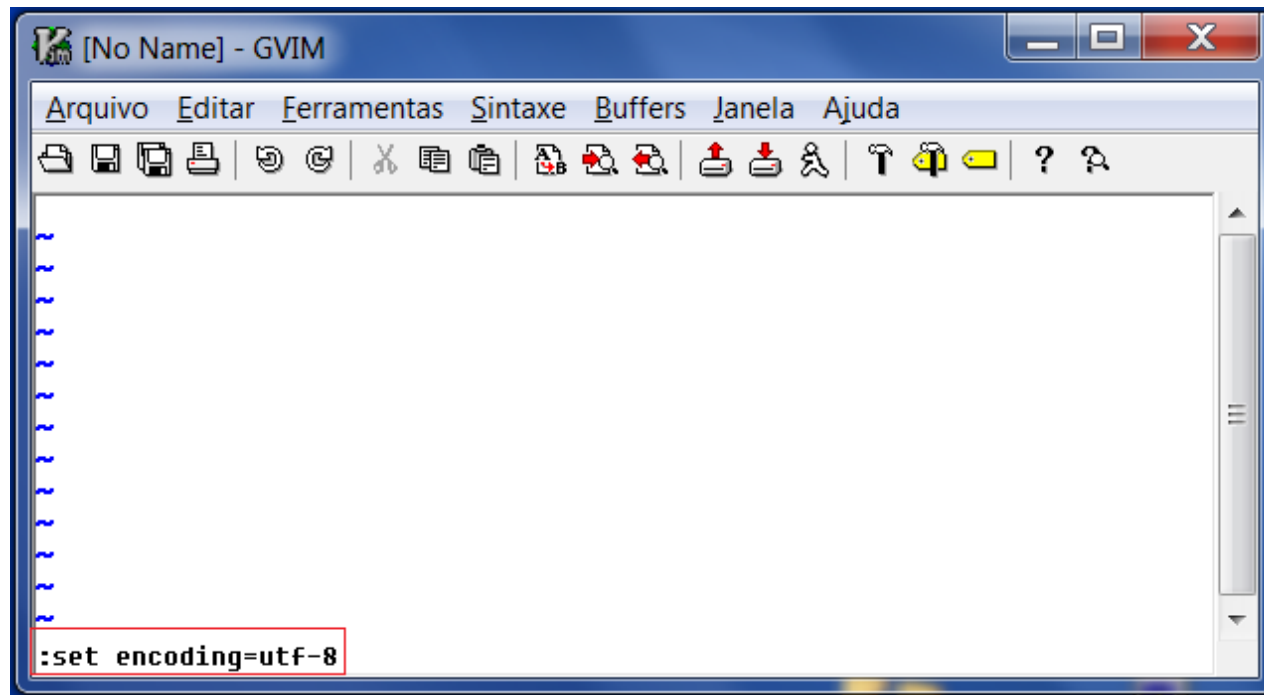
vim - UTF-8

- Verificando codificação atual



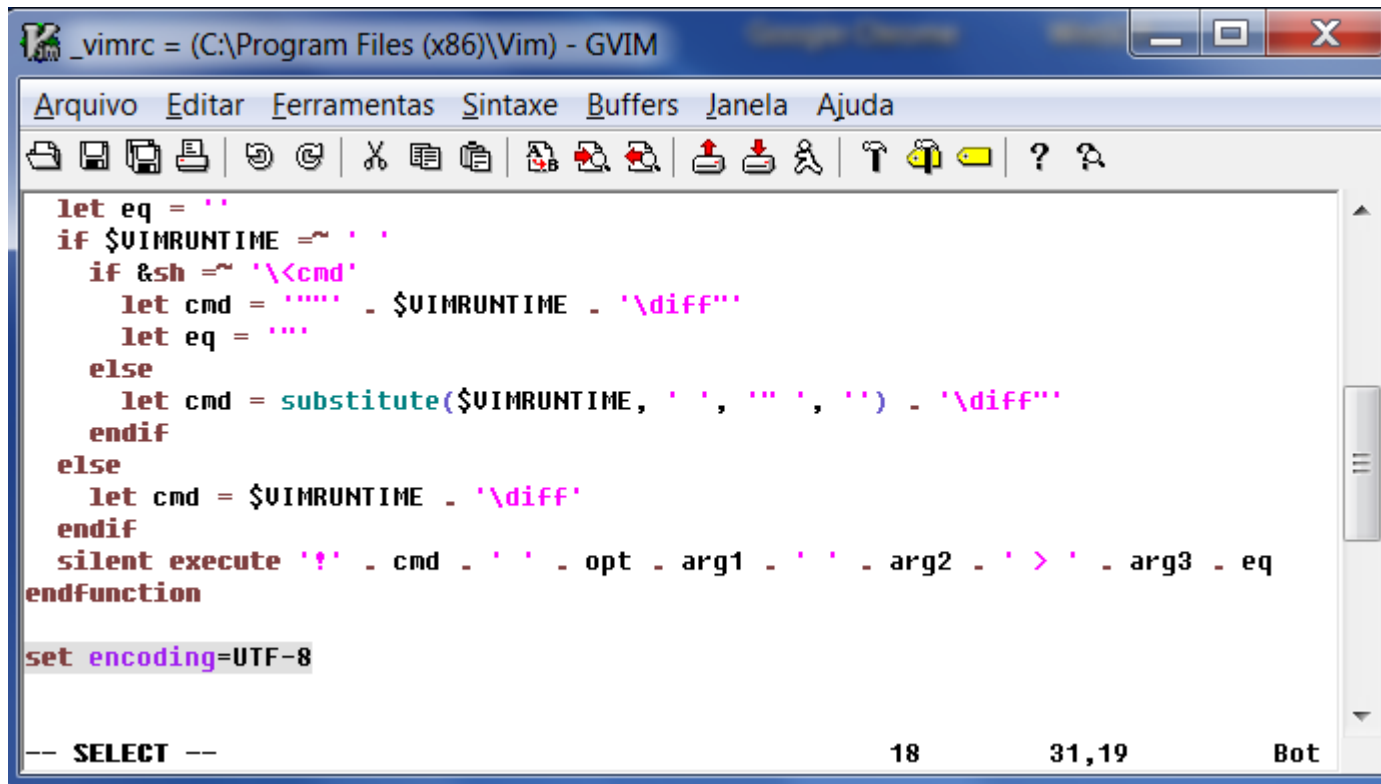
vim - UTF-8

- Alterando a codificação atual



vim - UTF-8

- Configurando no vimrc:



The screenshot shows the GVIM editor window titled "_vimrc = (C:\Program Files (x86)\Vim) - GVIM". The menu bar includes "Arquivo", "Editar", "Ferramentas", "Sintaxe", "Buffers", "Janela", and "Ajuda". The toolbar contains various icons for file operations and editing. The main text area displays the following Vim script code:

```
let eq = ''
if $VIMRUNTIME =~ ''
  if &sh =~ '\<cmd'
    let cmd = '""' . $VIMRUNTIME . '\diff'
    let eq = ''
  else
    let cmd = substitute($VIMRUNTIME, ' ', '" ', '') . '\diff'
  endif
else
  let cmd = $VIMRUNTIME . '\diff'
endif
silent execute '!' . cmd . ' ' . opt . arg1 . ' ' . arg2 . ' > ' . arg3 . eq
endfunction

set encoding=UTF-8
```

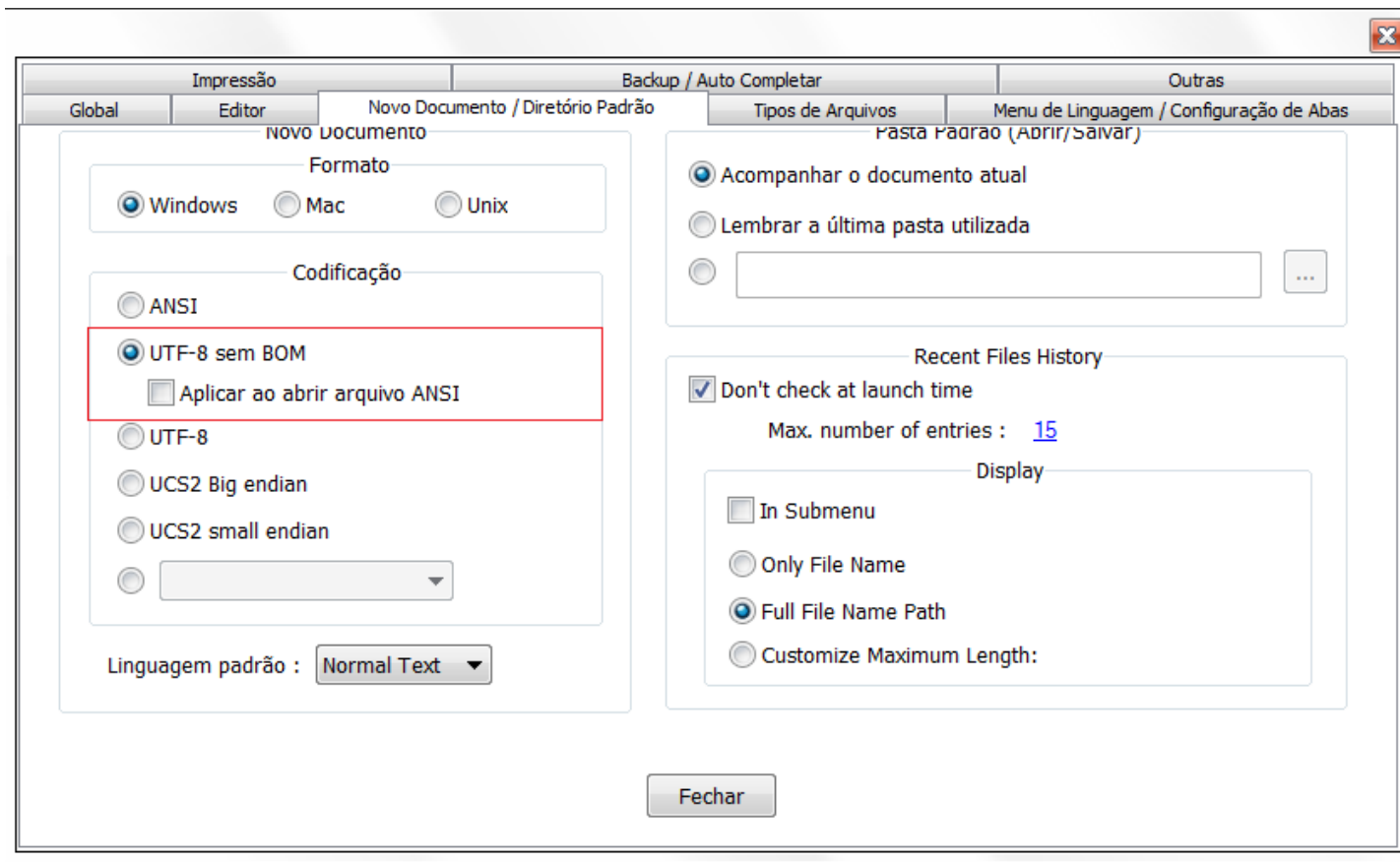
At the bottom of the window, the status bar shows "-- SELECT --" on the left, "18" in the center, "31,19" on the right, and "Bot" at the far right.

Tópicos

- Codificação de caracteres e Editores
 - Eclipse
 - Netbeans
 - vim
 - Notepad++

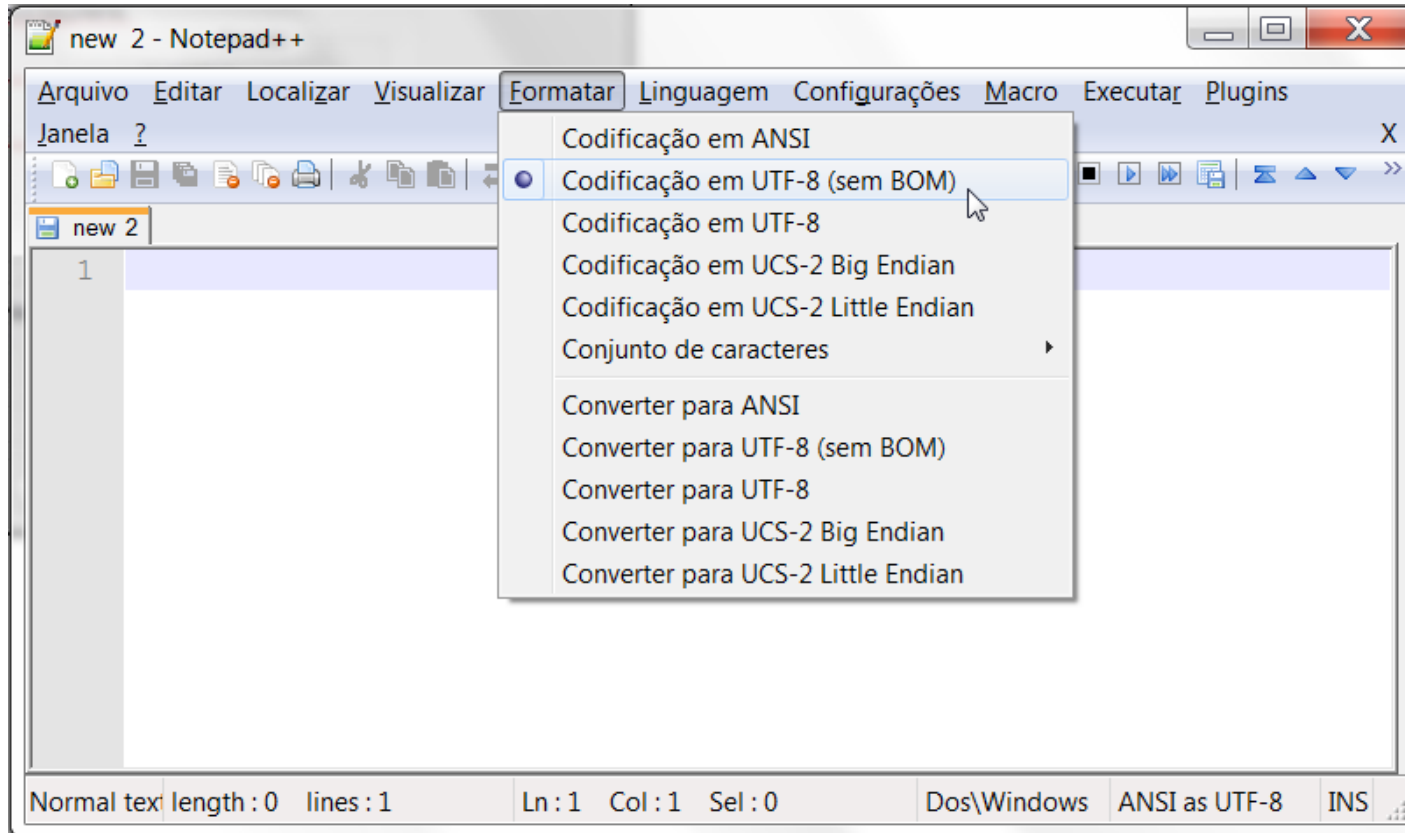
Notepad++ - UTF-8

- Configurações -> Preferências -> Novo Documento:



Notepad++ - UTF-8

- Arquivo aberto:



Indo Além

- Artigos

- <http://htmlpurifier.org/docs/enduser-utf8.html>
- <http://rubsphp.blogspot.com.br/search/label/charset>
- http://www.unicode.org/faq/utf_bom.html
- <http://webmonkeyuk.wordpress.com/2011/04/23/how-to-avoid-character-encoding-problems-in-php/>
- <http://docs.oracle.com/cd/E19253-01/817-2521/overview-207/index.html>
- <http://stackoverflow.com/questions/496321/utf8-utf16-and-utf32>
- <http://www.phpit.com.br/artigos/php-e-unicode-o-caminho-das-pedras.phpit>

Indo Além

- Tabelas Unicode

- <http://unicode-table.com/en/>
- <http://www.utf8-chartable.de/>
- <http://unicode.coeurlumiere.com/>

Indo Além

- PHP

- mbstring: <http://br1.php.net/mbstring>
- intl: <http://www.php.net/manual/en/book.intl.php>

Obrigado por Assistir



Twitter: @dpasqua

Blog: <http://douglaspasqua.com>

E-mail: douglas.pasqua@gmail.com

Facebook: <http://www.facebook.com/pasquablog>