Kapitel WT:II

II. Kommunikation und Protokolle für Web-Systeme

- Rechnernetze
- Prinzipien des Datenaustauschs
- Netzsoftware und Kommunikationsprotokolle
- Internetworking

- Client-Server-Interaktionsmodell
- Uniform Resource Locator

- Grundlagen HTTP-Protokoll
- Weitere HTTP-Konzepte
- Grundlagen TLS-Protokoll

- Zeichen und Codierung

Zeichen und Codierung

Konzepte

Unterscheidung folgender Konzepte [W3C] [unicode]:

1. abstraktes Zeichen (*abstract character, character*) [unicode 1 2]
   Element zur Konstruktion oder Repräsentation textueller Daten.

2. Zeichendarstellung (*glyph, glyph image, character shape*) [unicode 1, 2]

3. Zeichenvorrat, Zeichensatz (*character repertoire, character set*) [unicode 1, 2]
   Zusammengehörende Menge von abstrakten Zeichen (1) zur Repräsentation von Text.

Zeichen und Codierung

Konzepte

Unterscheidung folgender Konzepte [W3C] [unicode]:

1. abstraktes Zeichen (*abstract character, character*) [unicode 1 2]
   Element zur Konstruktion oder Repräsentation textueller Daten.

2. Zeichendarstellung (*glyph, glyph image, character shape*) [unicode 1, 2]

3. Zeichenvorrat, Zeichensatz (*character repertoire, character set*) [unicode 1, 2]
   Zusammengehörende Menge von abstrakten Zeichen (1) zur Repräsentation von Text.

4. Code-Raum (*codespace*) [unicode]
   Menge von Zahlen, die abstrakten Zeichen zugeordnet werden können. Zahlen des Code-Raums heißen Zeichencodes (*code points, character codes, character numbers*).

5. Code-Tabelle, codierter Zeichensatz (*coded character set, charset*) [unicode]
   Abbildung eines Zeichenvorrats bzw. Zeichensatzes (3) auf Zeichencodes (4).

6. Codierungsformat (*encoding form + encoding scheme, encoding*) [unicode 1, 2]
   Format der Byte-Repräsentation eines Zeichencodes (4).

Zeichen und Codierung

Konzepte (Fortsetzung)

Historisch liegen viele Code-Tabellen (5) in nur einem einzigen, „kanonischen“ Codierungsformat / Encoding (6) vor, in dem die Zeichencodes der Byte-Repräsentation entsprechen.

Beispielsweise werden der ASCII-Zeichensatz mit 7 Bit und, darauf aufbauend, viele westeuropäische Zeichensätze mit einem vollen Byte codiert.

➜ Raum für bis zu 128 bzw. 256 verschiedene Zeichen.

Sprache	(4) Code-Raum	(5) **Code-Tabelle** bzw. Charset	(6) **Codierungsformat** bzw. Encoding-Form	Code-Einheit	Länge
Englisch	0–7F	ASCII	*kanonisch*	7 Bit	7 Bit
Westeuropa	0–FF	ISO/IEC 8859-1	*kanonisch*	1 Byte	1 Byte
Chinesisch	0–FFFF	Big 5	*kanonisch*	2 Byte	2 Byte
Weltweit	0–10FFFF	Unicode	UTF-8 UTF-16 UTF-32	1 Byte 2 Byte 4 Byte	1–4 Byte 2 | 4 Byte 4 Byte

Zeichen und Codierung

Konzepte (Fortsetzung)

Auswahl druckbarer Zeichen der Code-Tabelle (5) ISO-8859-1 (Western Europe):

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
20	SP 0020	! 0021	" 0022	# 0023	$ 0024	% 0025	& 0026	' 0027	(0028	) 0029	* 002A	+ 002B	, 002C	- 002D	. 002E	/ 002F
30	0 0030	1 0031	2 0032	3 0033	4 0034	5 0035	6 0036	7 0037	8 0038	9 0039	: 003A	; 003B	< 003C	= 003D	> 003E	? 003F
40	@ 0040	A 0041	B 0042	C 0043	D 0044	E 0045	F 0046	G 0047	H 0048	I 0049	J 004A	K 004B	L 004C	M 004D	N 004E	O 004F
50	P 0050	Q 0051	R 0052	S 0053	T 0054	U 0055	V 0056	W 0057	X 0058	Y 0059	Z 005A	[005B	\ 005C	] 005D	^ 005E	_ 005F
60	` 0060	a 0061	b 0062	c 0063	d 0064	e 0065	f 0066	g 0067	h 0068	i 0069	j 006A	k 006B	l 006C	m 006D	n 006E	o 006F
70	p 0070	q 0071	r 0072	s 0073	t 0074	u 0075	v 0076	w 0077	x 0078	y 0079	z 007A	{ 007B	| 007C	} 007D	~ 007E	DEL 007F
A0	NBSP 00A0	¡ 00A1	¢ 00A2	£ 00A3	¤ 00A4	¥ 00A5	¦ 00A6	§ 00A7	¨ 00A8	© 00A9	ª 00AA	« 00AB	¬ 00AC	- 00AD	® 00AE	¯ 00AF
B0	° 00B0	± 00B1	² 00B2	³ 00B3	´ 00B4	µ 00B5	¶ 00B6	· 00B7	¸ 00B8	¹ 00B9	º 00BA	» 00BB	¼ 00BC	½ 00BD	¾ 00BE	¿ 00BF
C0	À 00C0	Á 00C1	Â 00C2	Ã 00C3	Ä 00C4	Å 00C5	Æ 00C6	Ç 00C7	È 00C8	É 00C9	Ê 00CA	Ë 00CB	Ì 00CC	Í 00CD	Î 00CE	Ï 00CF
D0	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß

[www.charset.org]

Bemerkungen:

- Ein Zeichenvorrat (3) für eine bestimmte Sprache heißt Alphabet oder Schrift. Beispiele: lateinische Schrift, chinesische Schrift.
- In der Praxis wird ASCII ebenfalls mit einem vollständigen Byte codiert, dessen MSB (most significant bit) stets null ist.
- Damit ein Parser ein Dokument lesen kann, muss er die verwendete Code-Tabelle / Charset (5) und dessen Codierungsformat / Encoding (6) kennen. [W3C]
- Ein Codierungsformat (6) (die Byte-Repräsentation eines Zeichencodes) heißt „kanonisch“, wenn der Zeichen*code* (4) eines Zeichens (1) auch als seine Byte-*Repräsentation* (6) verwendet wird.
- Unicode-Zeichencodes (*Code points*) (4) werden üblicherweise durch “U+” gefolgt von vier, fünf oder sechs hexadezimalen Ziffern dargestellt, zum Beispiel U+00AE oder U+1D310. [Wikipedia]
- Eine Anwendung kann ein Zeichen nur anzeigen, wenn sie auf eine Schriftart zugreifen kann, die eine Glyphe für dieses Zeichen enthält. [Wikipedia]

Bemerkungen (Zeichensätze Westeuropa) :

- Verschiedene Zeichensätze für westeuropäische Sprachen werden häufig synonym verwendet, obwohl sie sich in Details unterscheiden:
  - ISO/IEC 8859-1 (auch Latin-1) bezeichnet den ursprünglichen westlichen Zeichensatz, der in den Zeichencodes `20` – `7E` dem ASCII-Standard entspricht und in den Zeichencodes `A0` – `FF` weitere westeuropäische Zeichen codiert. Die Zeichencodes `00` – `1F` und `7F` – `9F` wurden für Steuerzeichen freigehalten. [www.charset.org]
  - ISO/IEC 8859-15 (auch Latin-9) ist eine Variante von ISO/IEC 8859-1, die das Währungszeichen ¤ durch das €-Symbol – sowie weitere selten verwendete Zeichen durch fehlende Sonderzeichen für Französisch, Estnisch und Finnisch ersetzt.
  - ISO-8859-1 und ISO-8859-15 („ISO-“ geschrieben) sind die IANA-Bezeichnungen für ISO/IEC 8859-1 und ISO/IEC 8859-15 mit C0- / C1-Steuerzeichen nach ISO/IEC 6429.
  - Windows-1252 (auch CP-1252, Code-Page 1252 oder ANSI) basiert auf ISO/IEC 8859-1 und füllt die Zeichencodes `00` – `1F` und `7F` mit Steuerzeichen und, bis auf 5 Ausnahmen, die Zeichencodes `80` – `9F` mit zusätzlichen druckbaren Zeichen wie dem €-Symbol.
  - Wegen häufig falscher Verwendung in der Vergangenheit behandelt der HTML5-Standard ISO-8859-1 bzw. Latin-1 als Windows-1252. [HTML5 § 4.2]
  - Die Unicode-Zeichencodes `U+0000` – `U+00FF` sind konsistent mit ISO-8859-1. UTF-8 codiert die ersten 128 dieser Zeichencodes kompatibel zu ASCII, wodurch ASCII eine Teilmenge von UTF-8 ist.

Zeichen und Codierung

Unicode

Die historischen, stark begrenzten Coderäume (4) sind nicht erweiterbar:

- ➜ Jede Sprachfamilie benötigt eigenen Zeichensatz (3) mit Code-Tabelle (5).
- ➜ Handhabung verschiedener Zeichensätze / Code-Tabellen ist fehleranfällig.
- ➜ Verschiedene Zeichensätze nur schwer im selben Dokument kombinierbar.

Zur Lösung dieses Problems wurde der Unicode-Standard entwickelt, der

1. einen einzigen, großzügig dimensionierten Coderaum spezifiziert und
2. Zeichen unabhängig von ihrer Byte-Repräsentation definiert.

Zeichen und Codierung

Unicode

Die historischen, stark begrenzten Coderäume (4) sind nicht erweiterbar:

- ➜ Jede Sprachfamilie benötigt eigenen Zeichensatz (3) mit Code-Tabelle (5).
- ➜ Handhabung verschiedener Zeichensätze / Code-Tabellen ist fehleranfällig.
- ➜ Verschiedene Zeichensätze nur schwer im selben Dokument kombinierbar.

Zur Lösung dieses Problems wurde der Unicode-Standard entwickelt, der

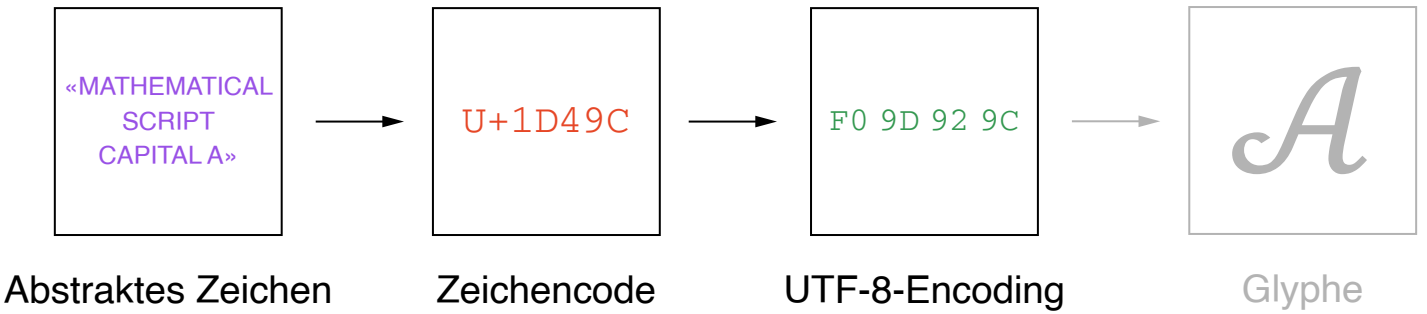
1. einen einzigen, großzügig dimensionierten Coderaum spezifiziert und
2. Zeichen unabhängig von ihrer Byte-Repräsentation definiert.

Anspruch von Unicode:

> "[...] *The Unicode Standard provides the capacity to encode all of the characters used for the written languages of the world. To keep character coding simple and efficient, the Unicode Standard assigns each character a unique numeric value and name.*" [unicode]

Zeichen und Codierung

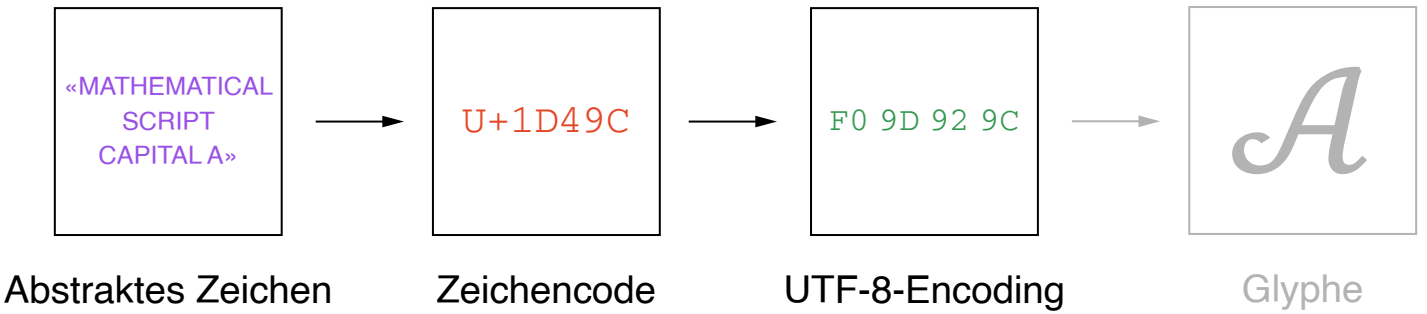
Unicode (Fortsetzung)



Der Unicode-Standard definiert für (fast) jedes auf der Welt verwendete Zeichen (1) einen eindeutigen Namen, einen Zeichencode (4), sowie drei Encoding-Formen (6).

Zeichen und Codierung

Unicode (Fortsetzung)

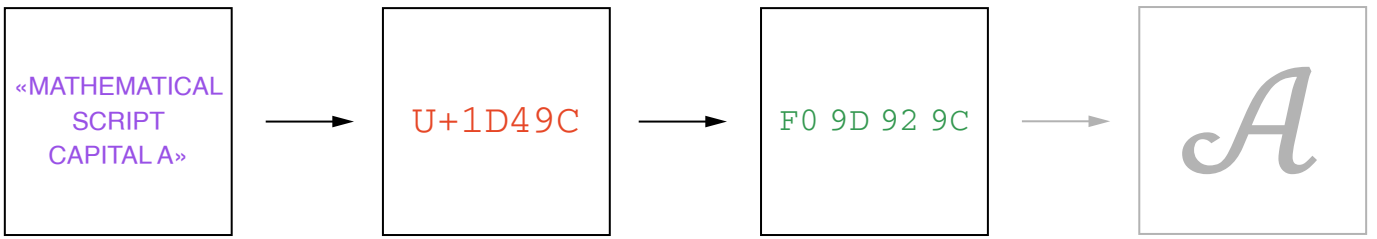


Der Unicode-Standard definiert für (fast) jedes auf der Welt verwendete Zeichen (1) einen eindeutigen Namen, einen Zeichencode (4), sowie drei Encoding-Formen (6).

- Die Zeichencodes sind in 17 aufeinander folgenden “Planes” zu je 65.536 (2^{16}, $FFFF_{16}$) Einträgen organisiert. Die meisten häufigen Zeichen passen in die erste Plane, auch *Basic Multilingual Plane* (BMP) genannt.
- Als Encodings stehen UTF-8, UTF-16 und UTF-32 mit unterschiedlich langen Codeworten bereit.

Zeichen und Codierung

Unicode (Fortsetzung)



Der Unicode-Standard definiert für (fast) jedes auf der Welt verwendete Zeichen (1) einen eindeutigen Namen, einen Zeichencode (4), sowie drei Encoding-Formen (6).

- Die Zeichencodes sind in 17 aufeinander folgenden "Planes" zu je 65.536 (2^{16}, $FFFF_{16}$) Einträgen organisiert. Die meisten häufigen Zeichen passen in die erste Plane, auch *Basic Multilingual Plane* (BMP) genannt.
- Als Encodings stehen UTF-8, UTF-16 und UTF-32 mit unterschiedlich langen Codeworten bereit.

Die Darstellung (*glyph images*) der abstrakten Zeichen wird explizit nicht definiert und ist Aufgabe des Ausgabegeräts. [unicode]

Zeichen und Codierung

Unicode: Encodings

UTF-8 (1 *bis* 4 Bytes) und UTF-16 (2 *oder* 4 Bytes) sind Codierungsformate mit variabler Länge. UTF-32 (4 Bytes) ist ein Codierungsformat mit fester Länge.

Zeichen und Codierung

Unicode: Encodings

UTF-8 (1 *bis* 4 Bytes) und UTF-16 (2 *oder* 4 Bytes) sind Codierungsformate mit variabler Länge. UTF-32 (4 Bytes) ist ein Codierungsformat mit fester Länge.

1. UTF-8-Codierung von Multibyte-Zeichen [RFC 3629]. Byte startet mit

0 :	ASCII-Zeichen.
$\{1\}_2^n\,0$:	erstes Byte einer Sequenz der Länge n, $n \in \{2,3,4\}$.
10 :	Folgebyte innerhalb einer Sequenz.

   Beispiele:

A	U+0041	$\rightsquigarrow$	0100 0001
Ä	U+00C4	$\rightsquigarrow$	1100 0011 1000 0100
$\mathcal{A}$	U+1D49C	$\rightsquigarrow$	1111 0000 1001 1101 1001 0010 1001 1100

   Sequenzlänge — Folgebytes

Zeichen und Codierung

Unicode: Encodings (Fortsetzung)

UTF-8 (1 *bis* 4 Bytes) und UTF-16 (2 *oder* 4 Bytes) sind Codierungsformate mit variabler Länge. UTF-32 (4 Bytes) ist ein Codierungsformat mit fester Länge.

2. UTF-16-Codierung von Multibyte-Zeichen [RFC 2781]:
   a) Zeichen mit Codes im Bereich `U+0000 – U+D7FF` und `U+E000 – U+FFFF` aus der BMP `(U+0000 – U+FFFF)` sind kanonisch codiert. ⇝ 63.487 Zeichen
   b) Weitere Zeichen sind als sogenannte Surrogat-Paare codiert, bestehend aus einem High surrogate `(U+D800 – U+DBFF)` und einem Low surrogate `(U+DC00 – U+DFFF)`. ⇝ ca. 1 Million Zeichen

Beispiele:

A	U+0041	⇝	0000 0000 0100 0001	(Big-Endian-Reihenfolge)
Ä	U+00C4	⇝	0000 0000 1100 0100	
$\mathcal{A}$	U+1D49C	⇝	1101 1000 0011 0101 1101 1100 1001 1100	

D 8 +1 D C

High surrogate Low surrogate

Zeichen und Codierung

Unicode: Encodings (Fortsetzung)

UTF-8 (1 *bis* 4 Bytes) und UTF-16 (2 *oder* 4 Bytes) sind Codierungsformate mit variabler Länge. UTF-32 (4 Bytes) ist ein Codierungsformat mit fester Länge.

3. UTF-32 (vormals UCS-4) codiert Unicode-Zeichen immer mit vier Bytes.

   Da es weniger als 2^{32} Unicode-Zeichen gibt (höchster definierter Zeichencode ist `U+10FFFF`), sind die ersten elf Bits immer null.

   Beispiele:

A	`U+0041`	$\rightsquigarrow$	`0000 0000 000`0 `0000 0000 0000 0000 0100 0001`
Ä	`U+00C4`	$\rightsquigarrow$	`0000 0000 000`0 `0000 0000 0000 0000 1100 0100`
$\mathcal{A}$	`U+1D49C`	$\rightsquigarrow$	`0000 0000 000`0 `0001 1101 0100 1001 1100`

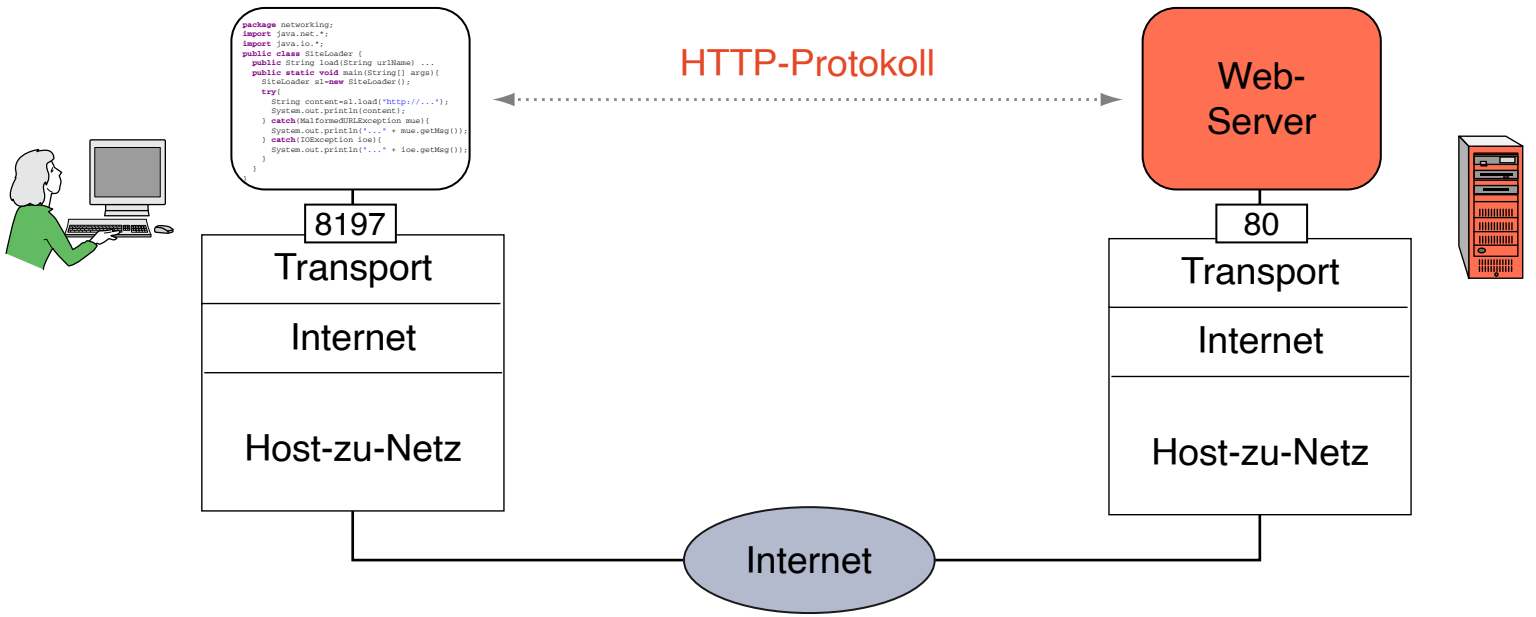
Bemerkungen:

- UTF steht für Unicode Transformation Format. [unicode]
- UTF-8 ist das am meisten verbreitete und allgemein vorzuziehende Unicode-Encoding.
- UTF-8 erfordert, Zeichen in der kürzest möglichen Form zu kodieren, auch wenn jedes Zeichen theoretisch mit führenden Nullen in vier Byte codiert werden könnte. D.h., „überlange“ Sequenzen sind nicht erlaubt. [RFC 3629]
- Die für UTF-16-Surrogate-Paare reservierten Zeichencodes sind einzeln keine gültigen Zeichen und stehen somit auch in UTF-8 und UTF-32 nicht zur Verfügung.
- Da Codeworte in UTF-16 und UTF-32 länger sind als ein Byte, wird am Anfang des Dokuments ein Byte Order Mark (BOM) zur Festlegung der Endianness benötigt.

  Deshalb wird beim Encoding (6) zwischen der Encoding-*Form* (UTF-8, UTF-16, UTF-32) und dem Encoding-*Schema* unterschieden, das auf der Encoding-Form aufbaut und die Byte-Reihenfolge (Endianness) hinzufügt.

Zeichen und Codierung

HTTP-Kommunikation [SiteLoader]



Zeichen und Codierung

HTTP-Kommunikation mit Java (Fortsetzung) [SiteLoader]

```
package documentlanguages.webcrawler;
import java.net.*;
import java.io.*;

public class SiteLoader2 {

    public static String load(String urlString) ...
    public static String extractCharset(String contentType, ...

    public static void main(String[] args){
        try{
            String content = SiteLoader2.load("http://www.heise.de");
            System.out.println(content);
        }
        catch(MalformedURLException e) {
            System.out.println("MalformedURLException:" + e.getMessage());
        }
        catch(IOException e) {
            System.out.println("IOException:" + e.getMessage());
        }
    }
}
```

Zeichen und Codierung

HTTP-Kommunikation mit Java (Fortsetzung) [SiteLoader]

```
public static String load(String urlString) throws IOException {
    URL url = new URL(urlString);
    HttpURLConnection con = (HttpURLConnection) url.openConnection();

    String contentType = con.getContentType();
    System.out.println("Content-Type: " + contentType);
    String encoding = extractCharset(contentType, "utf-8");
    System.out.println("Charset encoding: " + encoding);

    InputStream in = con.getInputStream();
    BufferedReader br = new BufferedReader(new InputStreamReader(in, encoding));
    String curline;
    StringBuilder content = new StringBuilder();

    while ((curline = br.readLine()) != null) {
        content.append(curline + '\n');
    }

    br.close();
    con.disconnect();
    return content.toString();
}
```

Zeichen und Codierung

HTTP-Kommunikation mit Java (Fortsetzung) [SiteLoader]

```
public static String extractCharset
    (String contentType, String defaultCharset) {
    // Extracts the charset value from the Content-Type header.
    // If no charset value is specified, the given default is returned.
    // Example. Content-Type: "text/html; charset=UTF-8"
    // Background: The *charset* value corresponds to the *encoding* if
    // for the charset only a single (canonical) encoding exists.

    String charset = defaultCharset;
    for (String param : contentType.replace(" ","").split(";")) {
        if (param.toLowerCase().startsWith("charset=")) {
            charset = param.split("=")[1];
            break;
        }
    }
    return charset;
}
```

Zeichen und Codierung

HTTP-Kommunikation mit Python [site_loader]

```
from urllib import request, error

def load(url):
    with request.urlopen(url) as stream:
        print(stream.code, stream.reason)
        encoding = stream.headers.get_content_charset()
        return stream.read().decode(encoding)

def main():
    try:
        content = load('https://heise.de')
        print(content)

    except error.HTTPError as e:
        print('HTTPError: ' + str(e))

    except error.URLError as e:
        print('URLError: ' + str(e))

if __name__ == '__main__':
    main()
```

Zeichen und Codierung

HTTP-Kommunikation mit Java / Python (Fortsetzung)

```
user@webis: bin$ java networkprotocol.SiteLoader2 | less

user@webis: python$ python3 site_loader2.py | less

Content-Type: text/html; charset=utf-8
Charset encoding: utf-8

<!DOCTYPE html>

<html lang="de">
<head>
<title>heise online - IT-News, Nachrichten und Hintergründe</title>
<meta name="description" content="News und Foren zu Computer, IT, Wissenschaft, ...
<meta name="keywords" content="heise online, c't, iX, Technology Review, ...
<meta name="publisher" content="Heise Zeitschriften Verlag" />
<meta name="viewport" content="width=1175" />
<link rel="home" type="text/html" title="Startseite" href="/" />
<link rel="copyright" title="Copyright" href="/impressum.html" />
<meta http-equiv="PICS-Label" content="(PICS-1.1 &quot;http://www.rsac.org/...
<script type="text/javascript" src="/js/jquery/jquery-1.7.1.min.js"></script>
...
</head>

<body>
...
```

Bemerkungen:

- Neben dem Entity-Header „Content-Type“ gibt es auch den Entity-Header „Content-Encoding“, der allerdings dafür verwendet wird, etwaige Komprimierungen des Entity-Body anzugeben (bspw. „gzip“).
- In der Meta-Information von HTML-Dokumenten verwendet man (abweichend von der Theorie) das Attribut `charset`, um das Codierungsformat / Encoding (6) (`UTF-8`, `UTF-16`, etc.) zu deklarieren – und nicht etwa die Code-Tabelle / Charset (5) (`Unicode`, etc.) [W3C]

  In dem Beispiel wird deshalb nicht das Encoding-Attribut (6), sondern das Charset-Attribut (5) abgefragt. Das macht dann keinen Unterschied, wenn das Encoding kanonisch ist, wie z.B. für den Charset `ISO-8859-1`.
- In der Meta-Information von XML-Dokumenten verwendet man (richtigerweise) das Attribut `encoding`, um das Codierungsformat / Encoding zu deklarieren. [W3C]
- Encoding-Deklarationen in einem XML- oder HTML-Dokument selbst sind nur mit ASCII-kompatiblen Encodings (z.B. UTF-8) möglich. Nicht-ASCII-kompatible Encodings müssen zwangsweise extern, etwa durch HTTP-Header, angegeben werden.

Zeichen und Codierung

Quellen zum Nachlernen und Nachschlagen im Web

- Radzivilovsky/Galka/Novgorodov. *UTF-8 Everywhere – Manifesto.* utf8everywhere.org
- Spolsky. *The Absolute Minimum Every Software Developer Must Know About Unicode.* www.joelonsoftware.com/articles/Unicode.htm
- Unicode. *Glossary.* www.unicode.org/glossary
- W3C. *Character Encodings for Beginners.* www.w3.org/International/questions/qa-what-is-encoding
- W3C. *Character encodings: Essential concepts.* www.w3.org/International/articles/definitions-character
- W3C. *Handling character encodings in HTML and CSS (tutorial).* www.w3.org/International/tutorials/tutorial-char-enc