Vorwissenschaftliche Arbeit

Programme und Fallbeispiele zur automatisierten Extraktion von Daten aus Webseiten

Verfasser

Rafael Vrečar, 8A

Schule

BRG19 Krottenbachstraße 11-13, 1190 Wien

Betreuer

Dipl.-Ing. Christian Schöbel

Abgabedatum

8. Februar 2016

Abstract

[Eine deutschsprachige Fassung des Abstracts ist im Anhang (A.I) angeführt.]

The automated extraction of data from web pages has the objective of saving time and avoiding mistakes. This prescientific paper is concerned with an automation process of this type.

The paper starts with the description of the proceedings which take place while surfing the web. Thereby, the Hypertext Transfer Protocol (HTTP) and its functional units client and server are being discussed. Furthermore, an empirical classification of the different kinds of web pages is taken. Subsequently, various tools which are necessary for this automation process are being examined. These are divided into the classes "Tools for retrieval" and "Tools for processing".

After the theoretical description of the tools the automation process is being analysed based on two case examples. In the first example the list of consultation hours of BRG19 is being extracted, whereas in the second example it is the individual list of modules which is being extracted. Both lists are being extracted from the downloaded files of the Hypertext Markup Language format (HTML) and converted into files of the Comma-separated values format (CSV).

Lastly, evaluations and conclusions concerning the automation process are being made. These considerations analyse which kind of knowledge is required to set up an automation of this kind and whether automation is reasonable or not.

Vorwort

Die Automatisierung von Arbeitsprozessen und die damit verbundene Effizienzmaximierung genoss bei mir schon lange reges Interesse, und da für mich schon vor dem Prozess der Themenfindung klar war, dass ich eine Arbeit mit informatischem Hintergrund schreiben möchte, war es für mich wenig überraschend, im Frühjahr 2015 folgende Themenstellung einzureichen:

> Programme und Fallbeispiele zur automatisierten Extraktion von Daten aus Webseiten.

An dieser Stelle möchte ich mich bei meiner Familie und meinen Freunden bedanken, die mich stets unterstützt haben. Der größte Dank gilt allerdings meinem Betreuer Dipl.-Ing. Christian Schöbel, der sich aller Fragen und Anliegen annahm, welche ich ihm im Schreibprozess dieser Arbeit entgegenbrachte.

Wien, 19. Jänner 2016

Rafael Vrečar

Wichtige Hinweise und Erklärungen

Im Vorfeld dieser vorwissenschaftlichen Arbeit sind einige Hinweise und Erklärungen anzustellen, um das Auftreten etwaiger Fragen, die innerhalb der eigentlichen Arbeit nicht beantwortet werden, zu vermeiden.

Hinweise zu Zitaten und Quellen(angaben)

In dieser vorwissenschaftlichen Arbeit befinden sich keine wörtlichen Zitate, da in einer technischen Arbeit, im Gegensatz zu einer geisteswissenschaftlichen Arbeit, in der Regel der Inhalt eines Texts, nicht aber dessen Wortlaut von Bedeutung ist. Sämtliche Quellen wurden grundsätzlich in der Form

> vgl. URL (Datum des letzten Zugriffs)
>
> (Anmerkung: weitere Spezifikation durch: „u. a.“, Kapitel- und Seitenangaben et cetera möglich)

in Fußnoten angegeben, da diese Form für eine auf Internetquellen basierende Arbeit als einzig sinnvolle erscheint. Diese Arbeit basiert zur Gänze auf Internetquellen, da die autoritativen Quellen ihre Inhalte in der Regel nicht in Papierform, sondern auf ihren Webseiten veröffentlichen. Besagte Quellen sind in einem Literaturverzeichnis (C Literaturverzeichnis) aufsteigend nach Fußnotennummern sortiert aufgelistet. Sie wurden im Zeitraum vom 19. bis zum 29. Jänner 2016 sorgfältig auf ihre Gültigkeit überprüft.

Hinweis zu Quellenangaben Glossarbegriffe betreffend

Im Verlauf dieser vorwissenschaftlichen Arbeit werden einige Begriffe auftreten, die in einem separaten Glossar erklärt sind. Diese sind entsprechend gekennzeichnet, eine genaue Erklärung diesbezüglich erfolgt noch in diesem Kapitel. Sofern Erklärungen schon im Text angestellt werden, da sie für das Verständnis der Arbeit erforderlich sind, werden die Quellen für die Erklärung allgemeiner Begriffe nicht nochmals angegeben. Besagte Quellen sind im Glossar angeführt. Der Text referenziert also auf das Glossar, das Glossar auf die entsprechende Quelle. Webseiten zu diversen spezifischen Begriffen, wie zum Beispiel Hypertext Transfer Protocol (HTTP), PHP Hypertext Processor (PHP) et cetera, werden hingegen bereits im Text als Fußnote angegeben.

Hinweis zum Abrufzeitpunkt der URLs von www.brg19.at

Diverse in den Fallbeispielen vorkommende URLs von www.brg19.at wurden allesamt, sofern nicht anders angegeben, am 20. Jänner 2016 auf ihre Gültigkeit überprüft. Änderun-

gen sind möglich und liegen nicht im Ermessen des Autors dieser Arbeit. Die auf dem beiliegenden Speichermedium vorhandenen Dateien wurden allesamt unter Absprache mit Dipl.-Ing. Christian Schöbel am 21. Jänner 2016 heruntergeladen.

Erklärungen zur Darstellung einzelner Textstücke

Im Folgenden werden Erklärungen zur Darstellung diverser Textstücke abgegeben. Dies betrifft Schriftart, Schriftgröße und Einrückung.

Erklärung zur Kennzeichnung von im Glossar enthaltenen Begriffen

Begriffe, die im Glossar enthalten sind, werden ab jetzt bei jeder Verwendung kursiv in der Schriftart Cambria dargestellt. Dabei sind auch Abwandlungen der Begriffe vollständig markiert. Zwischen Einzahl und Mehrzahl wird nicht unterschieden. Begriffe in Tabellen werden ebenfalls in dieser Form markiert.

Von dieser Markierung ausgeschlossen sind hingegen:

- Überschriften und im Text gesetzte Referenzen auf diese
- Bild-, Tabellen- und *Quelltext*unterschriften
- Begriffe in Grafiken
- Namen von Programmen, die im Rahmen dieser Arbeit als *Befehle* im *Terminal* ausgeführt wurden, sowie deren *Parameter*
- diverse Verzeichnisse

Abkürzungen sind ebenfalls im Glossar enthalten und werden in der gleichen Form dargestellt. Sie sind dort durch die Voranstellung von „Abk." entsprechend gekennzeichnet.

Erklärung zur Darstellung von URLs und Dateinamen

Vollständige *URLs*, also jene, die auch die Angabe des *Protokolls* enthalten, sowie Dateinamen, werden in dieser vorwissenschaftlichen Arbeit in der Schriftart `Consolas` dargestellt.

Erklärung zur Darstellung von Programmnamen, Befehlsketten und Quelltext

Die Namen von Programmen, die im Rahmen dieser Arbeit als *Befehle* im *Terminal* ausgeführt wurden, sowie deren *Parameter* sind ausnahmslos in der Schriftart `Consolas` dargestellt. Alle anderen Programme werden normal in der Schriftart des Texts dargestellt.

*Quelltext*beispiele sind ebenfalls ausnahmslos in der Schriftart `Consolas` dargestellt.

Erklärung zur Darstellung spezieller Zeichen

Zeichen sind dann in der Schriftart `Consolas` dargestellt, wenn sie sich auf einen ***Befehl*** beziehen.

Sofern Zeichen, wie zum Beispiel Bindestriche, ***Slashes*** und ähnliche, im Alltagsgebrauch auch in Texten vorkommen können und im Zusammenhang der Arbeit fälschlicherweise mit besagtem Gebrauch assoziiert werden könnten, sind sie unter Anführungszeichen gesetzt.

Erklärung zur Darstellung in verschiedenen Schriftgrößen

Ist es im Kontext dieser Arbeit erforderlich, dass ***Befehlsketten*** oder ***Zeichenketten*** in einer Zeile abgebildet werden, so ist die Schriftart, sofern notwendig, entsprechend verkleinert.

Erklärung zur Angabe von allgemeiner Syntax

Die allgemeine ***Syntax*** der einzelnen ***Tools*** hält sich an folgende Regeln:

- Die ***Zeichenfolge*** „…“ bedeutet, dass an dieser Stelle mehrere ***Optionen*** nebeneinander stehen können.
- ***Optionen***, die zwischen den Zeichen „`[`“ und „`]`“ und stehen, sind nicht zwingend erforderlich.

Erklärung zum Terminal

In das ***Terminal*** eingegebene ***Befehle*** werden immer auf jene Dateien angewendet, welche im ***Arbeitsverzeichnis*** abgelegt sind. In anderen ***Verzeichnissen*** abgelegte Dateien mit dem gleichen Dateinamen sind davon nicht betroffen. Es wurde gänzlich mit ***Bash***[1] gearbeitet.

Erklärung zur Entwicklungsumgebung des Quelltexts

Mit Ausnahme des ***Python***[2]***-Skripts*** im Anhang (A.III), welches unter ***Microsoft Windows XP***[3] erstellt wurde, wurden sämtliche Codezeilen dieser Arbeit unter ***Ubuntu Linux 14.04.3 LTS***[4] erstellt. Die Installation einzelner Programme ist eventuell notwendig, wird im Rahmen dieser vorwissenschaftlichen Arbeit jedoch nicht erläutert.

[1] vgl. https://www.gnu.org/software/bash/manual/bashref.html (19.01.2016)
[2] vgl. https://www.python.org/ (21.01.2016)
[3] vgl. http://windows.microsoft.com/de-de/windows/windows-help#windows=windows-xp (21.01.2016)
[4] vgl. http://releases.ubuntu.com/14.04/ (21.01.2016)

Inhaltsverzeichnis

1 Einleitung

Die Automatisierung bestimmter Arbeitsprozesse ist heutzutage essenziell, um ökonomisch mit Ressourcen haushalten zu können. Arbeitsprozesse, die automatisiert ablaufen, bedeuten eine große Zeit- und Geldersparnis. Die dadurch gewonnene Zeit kann bei Bedarf anderweitig verwendet werden. Das dadurch ersparte Geld eröffnet die Möglichkeit, in Güter oder Dienstleistungen zu investieren.

Diese vorwissenschaftliche Arbeit beschäftigt sich mit einem Teilbereich der Automatisierung, der automatisierten Extraktion von Daten aus ***Webseiten***.

Diese Arbeit ist als hybride Variante aus Literaturarbeit und empirischer Arbeit anzusehen. Der Literaturteil befasst sich mit dem Analysieren und Vergleichen der einzelnen ***Tools***. Der empirische Teil beschäftigt sich mit der Anwendung der einzelnen ***Tools*** im geeigneten Kontext. Dabei werden Fallbeispiele angeführt und praxisnahe in einem jeweils geeigneten Rahmen abgehandelt und analysiert.

Ferner werden das ***Internetsurfen*** im Allgemeinen und die verschiedenen Arten von ***Webseiten*** besprochen, wobei hier empirisch eine eigenständige Klassifizierung vorgenommen wird.

Ziel dieser Arbeit ist es, Schlussfolgerungen bezüglich der automatisierten Extraktion von Daten aus ***Webseiten*** zu ziehen. Dabei soll der gesamte Automatisierungsprozess beleuchtet werden. Die Weiterverarbeitung der gewonnenen Daten nach Abschluss des Extraktionsprozesses wird aufgrund des ansonsten überdimensionierten thematischen Rahmens nicht besprochen. Aus ebendiesem Grund ist es nicht Ziel dieser Arbeit, einen Überblick über die Entwicklungsgeschichte des ***HTTP*** oder jene der einzelnen ***Tools*** zu geben. Besagte Inhalte werden hinsichtlich jener Aspekte analysiert, die für die Arbeit relevant sind.

Als Quellen für diese Arbeit dienten hauptsächlich die offiziellen Dokumentationen und Spezifikationen der einzelnen ***Tools*** sowie die offiziellen ***Webseiten*** des ***World Wide Web Consortiums*** (***W3C***)[5] und der ***Internet Engineering Task Force*** (***IETF***)[6].

[5] vgl. http://www.w3.org/ (19.01.2016)
[6] vgl. https://www.ietf.org/ (19.01.2016)

2 Einführung in die Thematik

Im Vorfeld dieser vorwissenschaftlichen Arbeit ist es notwendig, einige Begriffe und Prozesse hinsichtlich ihrer Bedeutung für diese Thematik zu erklären. Das folgende Kapitel zielt darauf ab, den Gesamtprozess und die dabei vonstattengehenden Vorgänge vom manuellen Abrufen von ***Webseiten*** beim ***Internetsurfen*** bis hin zur automatisierten Extraktion von Daten aus diesen zu veranschaulichen.

2.1 Internetsurfen

Internetsurfen bezeichnet die Tätigkeit der wahllosen oder gezielten Suche von Informationen im Internet. Voraussetzung für das ***Internetsurfen*** ist, neben einer Internetverbindung, ein Programm, welches die Benutzerin beziehungsweise den Benutzer befähigt, dieser Tätigkeit nachzugehen. Dieses Programm ist der ***Webbrowser***, kurz ***Browser***. Beispiele für ***Browser*** sind ***Mozilla Firefox***[7], ***Google Chrome***[8], ***Windows Internet Explorer***[9], ***Microsoft Edge***[10], ***Safari***[11] und ***Opera***[12]. Der ***Browser*** muss gestartet werden, um mit dem ***Internetsurfen*** beginnen zu können.

Beim ***Internetsurfen*** werden ***Websites*** abgerufen. ***Websites*** bestehen aus einer oder mehreren ***Webseiten***.

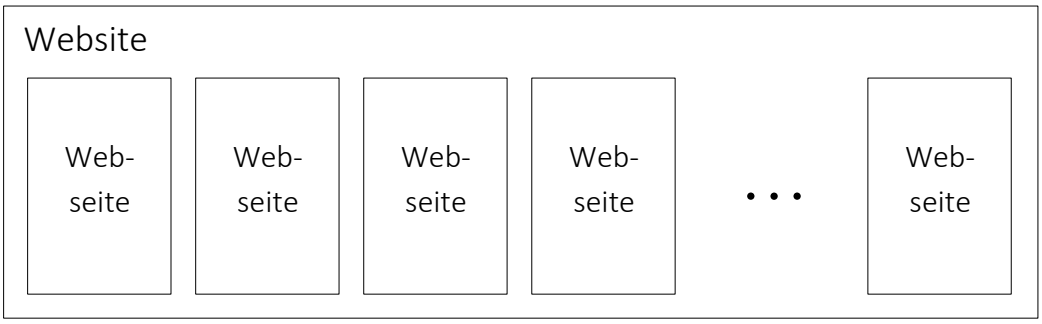

Abbildung 1: Eine Website besteht in der Regel aus mehreren Webseiten

[7] vgl. https://www.mozilla.org/de/firefox/new/ (19.01.2016)
[8] vgl. https://www.google.de/chrome/browser/desktop/index.html (19.01.2016)
[9] vgl. http://windows.microsoft.com/de-at/internet-explorer/download-ie (19.01.2016)
[10] vgl. https://www.microsoft.com/de-at/windows/microsoft-edge (19.01.2016)
[11] vgl. http://www.apple.com/at/safari/ (19.01.2016)
[12] vgl. http://www.opera.com/de (19.01.2016)

Da bei der automatisierten Extraktion von Daten aus ***Webseiten*** diese Unterscheidung allerdings irrelevant ist, da die Daten immer aus einzelnen ***Webseiten*** einer ***Website*** extrahiert werden, wird in dieser Arbeit lediglich der Begriff ***Webseite*** verwendet.

Um eine ***Webseite*** abzurufen, ist deren ***Adresse*** in die ***Adresszeile*** des ***Browsers*** einzugeben. Man kann aber auch nach einer ***Google-Suche***[13] auf eines der Ergebnisse klicken, um zu einer ***Webseite*** zu gelangen. Die entsprechende ***Webseite*** wird in beiden Fällen auf dem Bildschirm ausgegeben.

Die obig angesprochene ***Adresse*** hält sich an ein Schema, welches ***Uniform Resource Locator*** (***URL***)[14] genannt wird. Eine ***URL*** gibt an, welche Daten[15] von welchem Computer unter Verwendung des angegebenen ***Protokolls***[16] übertragen werden sollen, wobei hierbei unter ***Protokoll*** eine Kommunikationsvorschrift verstanden wird, die vorschreibt, wie eine Kommunikation abzulaufen hat.

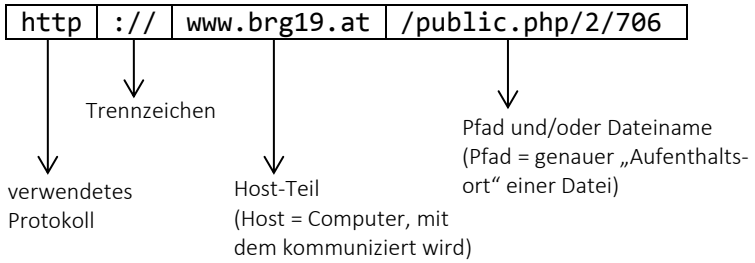

Abbildung 2: Ein konkretes Beispiel für eine URL

Gibt man keinen ***Pfad*** oder Dateinamen an, gelangt man je nach ***Server***einstellung zur Startseite.

2.1.1 Internetsurfen als Kommunikation

Abschließend ist zu sagen, dass das ***Internetsurfen*** als eine Kommunikation zwischen zwei Instanzen anzusehen ist: der Person, die eine ***Webseite*** abrufen will, und jener, die diese anbietet. Diese Darstellung ist jedoch vereinfacht und wird im Folgenden konkretisiert.

[13] vgl. https://www.google.com (19.01.2016)
[14] vgl. https://tools.ietf.org/html/rfc1738 (19.01.2016)
[15] vgl. https://tools.ietf.org/html/rfc1738#section-2 (19.01.2016)
[16] vgl. https://tools.ietf.org/html/rfc1738#section-2.1 (19.01.2016)

2.2 Transport von Webseiten gemäß HTTP

Die obig angesprochene Kommunikation läuft, wenn in der *URL* `http://` vorangestellt ist, gemäß dem *Hypertext Transfer Protocol* (*HTTP*) ab.

Das *HTTP* ist keineswegs das einzige *Protokoll*, mit dem Inhalte über das Internet übertragen werden können. Weitere Beispiele für *Protokolle* sind das *File Transfer Protocol* (*FTP*)[17], das *Simple Mail Transfer Protocol* (*SMTP*)[18], das *Post Office Protocol* (*POP*)[19], das *Internet Message Access Protocol* (*IMAP*)[20], das *Lightweight Directory Access Protocol* (*LDAP*)[21] und das *Network Time Protocol* (*NTP*)[22], um nur einige zu nennen.

Aufgrund seiner weiten Verbreitung beim *Internetsurfen* ist das *HTTP* allerdings das für diese Arbeit relevante *Protokoll*.

Das *HTTP* ist ein *zustandsloses Protokoll* zur Übertragung von Daten über ein *Rechnernetz*. *Zustandslos* bedeutet in diesem Fall, dass alle *Anfragen* voneinander unabhängig behandelt werden.[23]

2.2.1 Funktionseinheiten des HTTP

Das *HTTP* hat verschiedene Funktionseinheiten, welche im *Request for Comments* (*RFC*) 2616 in Abschnitt 1.3[24] spezifiziert sind.

Eine Kommunikation gemäß dem *HTTP* läuft immer zumindest zwischen den zwei Funktionseinheiten *Client* und *Webserver*, oft einfach *Server*, ab:

- Der *Client* initiiert Verbindungen zu *Servern* und ist in der Regel ein *Browser*.
- Der *Server* wartet auf Verbindungs*anfragen* von *Clients* und antwortet auf diese durch das Liefern der gewünschten Ressource oder Statusinformation.

[17] vgl. https://tools.ietf.org/html/rfc354 (19.01.2016)
[18] vgl. https://tools.ietf.org/html/rfc5321 (19.01.2016)
[19] vgl. https://tools.ietf.org/html/rfc1939 (19.01.2016)
[20] vgl. https://tools.ietf.org/html/rfc3501 (19.01.2016)
[21] vgl. https://tools.ietf.org/html/rfc4511 (19.01.2016)
[22] vgl. https://tools.ietf.org/html/rfc5905 (19.01.2016)
[23] vgl. https://tools.ietf.org/html/rfc2616, Abstract (19.01.2016)
[24] vgl. http://tools.ietf.org/html/rfc2616#section-1.3 (19.01.2016)

Folgende Grafik veranschaulicht den Ablauf einer Kommunikation gemäß dem *HTTP* auf Basis der genannten Funktionseinheiten:

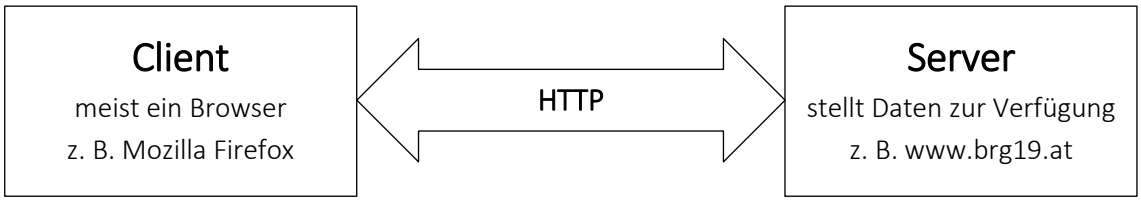

Abbildung 3: Kommunikation zwischen Client und Server gemäß HTTP (1)

2.2.2 Kommunikationsablauf aus Client-Sicht

Die Kommunikation läuft bei einer *HTTP*-Verbindung immer nach einem bestimmten Schema ab. Aus *Client*-Sicht lässt sich der typische Kommunikationsablauf wie folgt aufschlüsseln:

(1) Der *Client* öffnet eine Verbindung[25] zum *Server*.
(2) *HTTP-Request*: Der *Client* sendet eine *Anfrage* an den *Server*.
(3) Der *Server* erhält die *Anfrage*.
(4) *HTTP-Response*: Der *Server* antwortet auf die *Anfrage*.
(5) [Beenden der Verbindung]

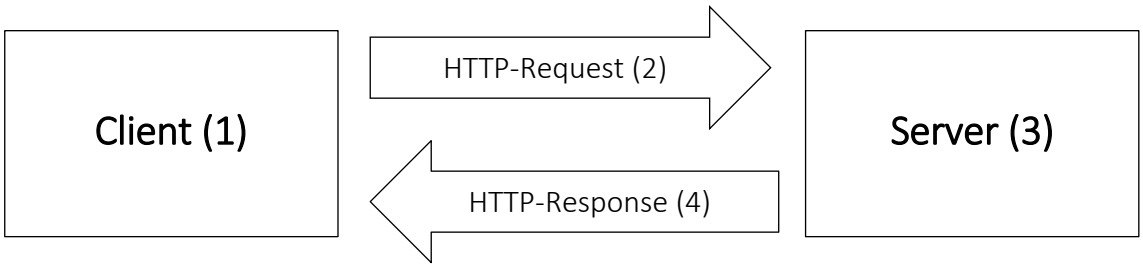

Abbildung 4: Kommunikation zwischen Client und Server gemäß HTTP (2)

2.2.3 HTTP-Request

Ein *Request* (*Anfrage*) wird in Form einer *Request-Message* [26] (*Anfragenachricht*) vom *Client* zum *Server* ausgeschickt. In der ersten Zeile der Nachricht sind die auf die Ressource

[25] vgl. https://www.rfc-editor.org/rfc/rfc793.txt (19.01.2016)
[26] vgl. http://www.w3.org/Protocols/rfc2616/rfc2616-sec5.html (19.01.2016)

angewendete Methode[27], der Bezeichner der Ressource (*Request-URL*[28]) und die *Protokoll*version angeführt.

GET	/index.php	HTTP/1.1
Methode	Request-URL	Protokollversion

Abbildung 5: Ein konkretes Beispiel für eine Request-Zeile

Das *HTTP* bietet verschiedene Methoden einer *Request-Message* an. Für die automatisierte Datenextraktion sind in erster Linie folgende relevant:

GET[29]: Diese Methode sendet eine *Anfrage* nach der in der *Request-URL* angegebenen Ressource. *Client*-Daten werden als Bestandteil der *URL* der Ressource übergeben. Diese Methode wird für jene Informationen verwendet, die offenkundig in der *URL* stehen dürfen. Außerdem wird der Benutzerin beziehungsweise dem Benutzer auf diese Art und Weise die Möglichkeit geboten, *Parameter* direkt in der *Adresszeile* des *Browsers* in der *URL* auszubessern.

POST[30]: Diese Methode funktioniert wie *GET*, allerdings werden dabei *Client*-Daten nicht an die *URL* angehängt, sondern im Verborgenen übermittelt. Diese Methode ist für sensible Informationen wie zum Beispiel Passwörter wichtig.

2.2.4 HTTP-Response

Nach dem Erhalten und Interpretieren einer *Request-Message* antwortet der *Server* mit einem *Response* (*Antwort*) in Form einer *Response-Message*[31] (*Antwortnachricht*).

Die erste Zeile einer *Antwortnachricht* ist die Statuszeile, bestehend aus der *Protokoll*version, gefolgt von einem numerischen Statuscode und der ihm zugeordneten Phrase.[32]

HTTP/1.1	200	OK
Protokollversion	numerischer Statuscode[33]	zugehörige Phrase[34]

Abbildung 6: Ein konkretes Beispiel für eine Statuszeile

[27] vgl. http://tools.ietf.org/html/rfc2616#section-5.1.1 (19.01.2016)
[28] vgl. http://tools.ietf.org/html/rfc3986#section-1.1.3 (19.01.2016)
[29] vgl. http://tools.ietf.org/html/rfc2616#section-9.3 (19.01.2016)
[30] vgl. http://tools.ietf.org/html/rfc2616#section-9.5 (19.01.2016)
[31] vgl. http://www.w3.org/Protocols/rfc2616/rfc2616-sec6.html (19.01.2016)
[32] vgl. http://www.w3.org/Protocols/rfc2616/rfc2616-sec6.html (19.01.2016)
[33] vgl. http://www.iana.org/assignments/http-status-codes/http-status-codes.xhtml (19.01.2016)
[34] vgl. http://www.iana.org/assignments/http-status-codes/http-status-codes.xhtml (19.01.2016)

2.2.5 HTTPS

Die verschlüsselte Variante des *HTTP* heißt *HTTPS*, wobei das „S" für „secure" (sicher) steht. Diese ist im *RFC* 2818[35] spezifiziert. Durch *HTTPS* ist eine sichere Übertragung von sensiblen Daten[36] wie beispielsweise Passwörtern möglich.

2.2.6 Session-Management

Da das *HTTP*, wie in der Einleitung bereits erwähnt, ein *zustandsloses Protokoll* ist, bezieht es keine Information aus einer Kommunikation, die bereits in der Vergangenheit stattgefunden hat.[37]

In diesem Zusammenhang ist der Begriff *Session* (*Sitzung*) zu erwähnen. Eine *Session* beschreibt einen Dialog, welcher sich über mehrere Zyklen von *Request* und *Response* erstreckt. Weiter soll innerhalb einer *Session* der Rückbezug auf die vergangene Kommunikation möglich sein.[38]

Zur Codierung von *Session*-Information werden verschiedene Techniken[39] verwendet, die das *HTTP* unterstützt:

- Die *Session*-Information wird beim *URL Rewriting* in der *URL* codiert.
- Die *Session*-Information wird in Form von *Cookies*[40] beim *Client* gespeichert.
- Die *Session*-Information wird in sogenannten *Hidden Fields*, unsichtbaren Formularfeldern, untergebracht.

2.2.7 Content-Type / MIME-Type

Unter Verwendung des *Protokolls HTTP* können verschiedene Typen von Daten übertragen werden. Die sogenannten *Content-* beziehungsweise *MIME-Types* (*Medientypen*)[41] geben an, um welche Art von Daten es sich handelt. Sie bestehen aus einem *Top-Level-Typ* und einem *Untertyp* und können durch *Parameter* weiter spezifiziert werden.

---

[35] vgl. https://tools.ietf.org/html/rfc2818 (19.01.2016)
[36] vgl. https://tools.ietf.org/html/rfc2818#section-1 (19.01.2016)
[37] vgl. http://tools.ietf.org/html/rfc7230#section-2.3 (19.01.2016)
[38] vgl. http://www.uni-weimar.de/medien/webis/teaching/lecturenotes/web-technology/unit-de-network-protocol3.pdf, Seite 40 (19.01.2016)
[39] vgl. http://www.uni-weimar.de/medien/webis/teaching/lecturenotes/web-technology/unit-de-network-protocol3.pdf, Seite 42 (19.01.2016)
[40] vgl. https://tools.ietf.org/html/rfc6265 (19.01.2016)
[41] vgl. http://www.w3.org/2001/tag/2002/0129-mime (19.01.2016)

Folgende Tabelle veranschaulicht einige Arten der übertragenen Daten:

Top-Level-Typ	Untertyp	Beschreibung
text	*plain* *enriched*	unformatierter *ASCII-Text* *ASCII-Text* mit einfachen Formatierungen
image	gif jpeg	Standbild im *GIF-Format* Standbild im *JPEG-Format*
audio	basic	Klangdaten
application	postscript	druckbares Dokument im *PostScript-Format*
…	…	…

Tabelle 1: Auszug aus einer Übersicht der erlaubten MIME-Types[42]

2.3 Woraus Webseiten bestehen

Da gemäß dem *HTTP* verschiedene *MIME-Types* übertragen werden können, steht der eine *Webseite* anbietenden Person ein breites Feld an Möglichkeiten zur Verfügung, Daten bereitzustellen.

2.3.1 Hypertext Markup Language (HTML)

Sehr viele *Webseiten* sind aus *Hypertext* aufgebaut. *Hypertext* ist ein über *Links* verbundenes Netz aus Text-, Bild- und Dateneinheiten. *Hypertext* wird durch die *Hypertext Markup Language* (*HTML*) ausgezeichnet.

HTML ist eine Auszeichnungssprache zum Beschreiben der Struktur von *Webseiten*. *HTML* ermöglicht Entwicklerinnen und Entwicklern Online-Dokumente mit Überschriften, Text, Tabellen, Listen, Fotos, Formularen, Videoclips, Soundclips et cetera zu veröffentlichen. Weiter ermöglicht *HTML* durch klickbare *Links* zwischen einzelnen *Webseiten* zu navigieren.[43]

[42] vgl. http://www.iana.org/assignments/media-types/media-types.xhtml (19.01.2016)
[43] vgl. http://www.w3.org/standards/webdesign/htmlcss#whathtml (19.01.2016)

Ein Beispiel für ein einfaches *HTML*-Dokument, zeilenweise aufgegliedert:

```
1  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
2  <html>
3  <head>
4    <title>Titel der Webseite</title>
5  </head>
6  <body>
7    <h1>Überschrift erster Ebene</h1>
8    <p>einfacher Text</p>
9  </body>
10 </html>
```

Quelltext 1: Aufbau eines einfachen HTML-Dokuments

Ein *HTML*-Dokument besteht aus drei Teilen:[44]

1. einer Zeile, die Informationen über die *HTML*-Version enthält (Zeile 1)
2. dem *HTML*-Kopf, der Informationen über das Dokument, sogenannte ***Metadaten***, bereitstellt, die in der Regel nicht dargestellt werden (Zeilen 3 bis 5)
3. dem *HTML*-Körper, der den Inhalt enthält, welcher dargestellt wird (Zeilen 6 bis 9)

Der *HTML*-Kopf und der *HTML*-Körper sind von den ***Tags*** `<html>` und `</html>` umschlossen.

Tags zeichnen Elemente aus. Sie können Elemente beispielsweise als Überschrift oder ***Link*** auszeichnen. Hierbei wird zwischen öffnenden und schließenden ***Tags*** unterschieden:

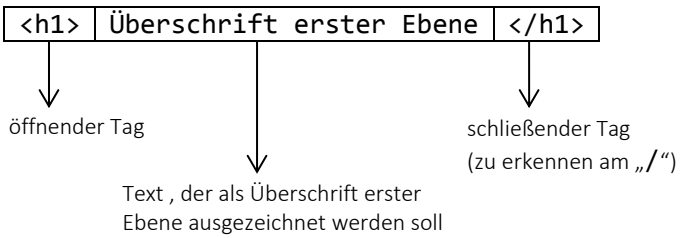

Abbildung 7: Öffnende und schließende Tags in HTML

Diese ***Tags*** lassen sich nach der entsprechenden ***Dokumenttyp***definition verschachteln[45], dadurch wird beispielsweise die Ver***link***ung einer Überschrift ermöglicht:

```
<h1><a href="Zieladresse">Überschrift</a></h1>
```

[44] vgl. http://www.w3.org/TR/html401/struct/global.html#h-7.1 (19.01.2016)
[45] vgl. http://www.w3.org/TR/html401/sgml/dtd.html (19.01.2016)

Innerhalb der *Tags* `<h1>` und `</h1>` befindet sich jener Inhalt, der als Überschrift formatiert werden soll. Innerhalb von `<a>` und `</a>` steht der *Link*text, also jener Text, der auf dem Bildschirm dargestellt wird, wobei `href` ein *Parameter* des *Link-Tags* ist, der die Ziel*adresse* angibt.

2.3.2 Cascading Style Sheets (CSS)

Während *HTML* die Struktur einer *Webseite* deklariert, gibt *Cascading Style Sheets* (*CSS*) die Darstellung einer *Webseite* an.[46]

Durch *CSS* können Schriftart, Größe und Farbe eines Texts verändert, sowie sonstige Formatierungen vorgenommen werden.[47]

CSS kann direkt in einem *HTML*-Dokument verwendet, oder in ein separates Dokument ausgelagert werden.[48]

Der Unterschied zwischen HTML und *CSS* ist wie folgt zu veranschaulichen:

Struktur	Darstellung
HTML	*CSS*
Das Wort „VwA“ ist eine Überschrift	Überschriften sind rot

Tabelle 2: HTML und CSS haben unterschiedliche Aufgaben

Sofern keine *CSS*-Informationen angegeben beziehungsweise verfügbar sind, wird eine *Webseite* wie in den Einstellungen des *Browsers* festgelegt dargestellt.[49]

2.3.3 JavaScript

HTML und *CSS* sind keine Programmiersprachen, *JavaScript* ist eine. *JavaScript* bietet die Möglichkeit, Webanwendungen zu realisieren. Dies ist durch *HTML* und *CSS* allein nicht möglich. Viele *Browser* haben ihre eigene *JavaScript*-Spezifikation[50], daher kann es zu Kompatibilitätsproblemen kommen. Der Sprachkern wurde von der *Ecma International* (*Ecma*) in der *ECMA* 262 als *ECMAScript* spezifiziert.[51]

[46] vgl. http://www.w3.org/standards/webdesign/htmlcss#whatcss (19.01.2016)
[47] vgl. http://www.w3.org/TR/CSS1/#abstract (19.01.2016)
[48] vgl. http://www.w3.org/TR/CSS21/intro.html#html-tutorial (19.01.2016)
[49] vgl. u. a. https://support.mozilla.org/de/kb/Einstellungen-Fenster--Inhalts-Abschnitt (19.01.2016)
[50] vgl. u. a. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference (19.01.2016)
[51] vgl. http://ecma-international.org/ecma-262/5.1/ (19.01.2016)

2.4 Manueller Abruf und manuelle Datenextraktion

Damit ist geklärt, was beim ***Internetsurfen*** passiert, sprich wie die Kommunikation zwischen ***Client*** und ***Server*** abläuft, wie ***Webseiten*** transportiert werden und woraus sie bestehen. Dies hat allerdings grundsätzlich noch nicht direkt etwas mit Automatisierung zu tun, da es beim ***Internetsurfen*** immer eine Person ist, welche die ***URL*** in die ***Adresszeile*** des ***Browsers*** eingibt oder auf ein Suchergebnis klickt. Das Abrufen einer ***Webseite*** geht also manuell vonstatten. Ein Beispiel hierfür ist das manuelle Abrufen von Börsenkursen oder Sportergebnissen. Eine Person besucht eine ***Webseite***, um diese Information zu erhalten und bei Bedarf zu extrahieren. Die Extraktion der Daten erfolgt ebenfalls auf manuellem Wege, beispielsweise durch das Aufschreiben auf ein Blatt Papier oder durch die Eingabe in beispielsweise ein ***Microsoft Excel***[52] Dokument.

Aufgrund des hohen Zeitaufwandes, der Fehleranfälligkeit oder anderer Faktoren kann diese Art der Datenextraktion auf Dauer auf vielen Ebenen eventuell nicht rentabel sein und den Bedarf einer Automatisierung hervorrufen.

2.5 Automatisierter Abruf und automatisierte Datenextraktion

Um den obig angesprochenen Prozess zu automatisieren, müssen sowohl Abruf als auch Datenextraktion automatisiert werden. Der Gesamtprozess dieser Automatisierung sieht hierbei wie folgt aus („3. Datenverarbeitung“ wird, wie in der Einleitung bereits erwähnt, im Rahmen dieser Arbeit nicht behandelt):

Abbildung 8: Gesamtprozesses der automatisierten Datenextraktion aus Webseiten

Für den automatisierten Abruf und die automatisierte Datenextraktion werden ***Werkzeuge***, auch ***Tools*** genannt, benötigt.

Das ***Tool*** zum Abruf kann entweder der ***Browser*** selbst oder ein speziell dafür angefertigtes sein. Von den bereits angesprochenen ***Browsern*** gibt es zwei verschiedene Arten: ***Browser*** mit grafischer Benutzeroberfläche und jene, die textbasiert sind. Die in Kapitel 2.1 bereits

[52] vgl. https://products.office.com/de-at/excel (19.01.2016)

genannten sind allesamt *Browser* mit grafischer Benutzeroberfläche. *Browser* mit grafischer Benutzeroberfläche stellen eine *Webseite* grafisch dar, textbasierte in Textform. Ein Beispiel für einen textbasierten *Browser* ist `lynx`[53].

2.6 Reflexion über das Thema Webseiten und daraus zu ziehende Schlüsse

Bei der Reflexion über das Thema *Webseiten* kommt man zum Schluss, dass diese auf interaktive Bedienung und nicht auf die automatische Verarbeitung, welche eine nicht interaktive Anwendung darstellt, ausgelegt sind. *Webseiten* haben Bedienelemente wie Navigationsleisten und Menüs, eine entsprechende grafische Aufbereitung und ähnliche Elemente, die mit der eigentlichen Information, welche die *Webseite* transportieren möchte, teilweise wenig zu tun haben.

Browser als Mittel zum *Internetsurfen* sind demnach auch primär auf interaktive Bedienung ausgelegt, weshalb der automatische Abruf der Daten ohne das Zutun einer Benutzerin beziehungsweise eines Benutzers kein vorgesehener Anwendungsfall ist.

Der textbasierte *Browser* eröffnet in diesem Zusammenhang eine Möglichkeit, da er die Daten in Textform extrahieren kann. Diese Vorgehensweise klappt allerdings bei unterschiedlichen *Webseiten* unterschiedlich gut. Wie performant diese Methode ist, hängt von der Technologie, die hinter einer *Webseite* steckt, ab. Andere *Tools* verhalten sich ähnlich. *Webseiten* sind also bezüglich ihres technologischen Hintergrunds, der für den *Client* zumindest teilweise sichtbar ist, einzuteilen.

[53] vgl. http://lynx.invisible-island.net/ (19.01.2016)

3 Klassifizierung von Webseiten

Die Technologie von ***Webseiten*** betreffend werden verschiedene Entwicklungsparadigmen verfolgt, bezüglich welcher ***Webseiten*** im Folgenden klassifiziert werden.

Die in dieser Arbeit getroffene Klassifizierung sieht wie folgt aus:

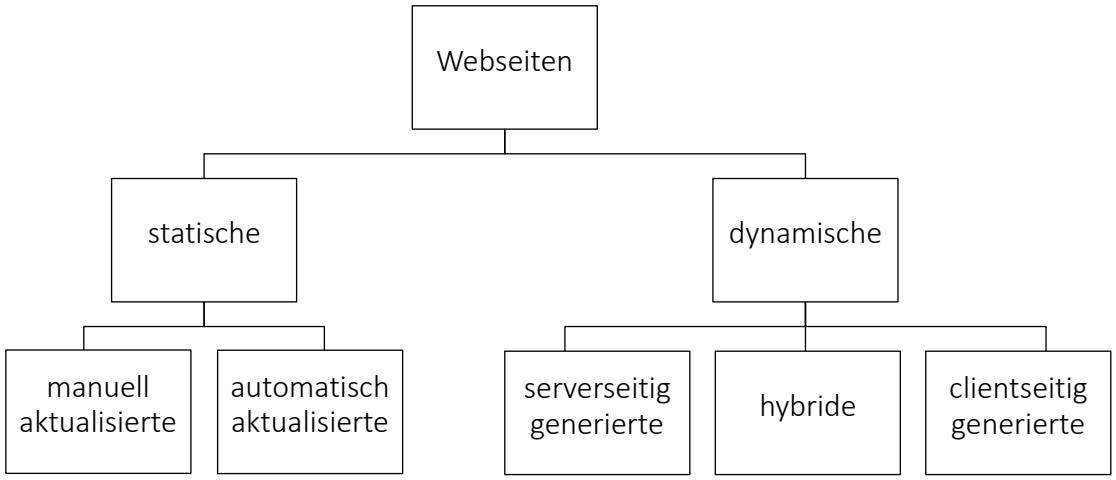

Abbildung 9: Einteilung der Arten von Webseiten

Die erste Unterscheidung, die getroffen werden muss, ist somit jene zwischen ***statischen*** und ***dynamischen Webseiten***:

3.1 Statische Webseiten

Statische Webseiten zeichnen sich dadurch aus, dass der ***Client*** beim Abruf exakt die gleiche Datei erhält, die auf dem ***Server*** gespeichert ist. Der ***Client*** holt also statische Dateien ab.

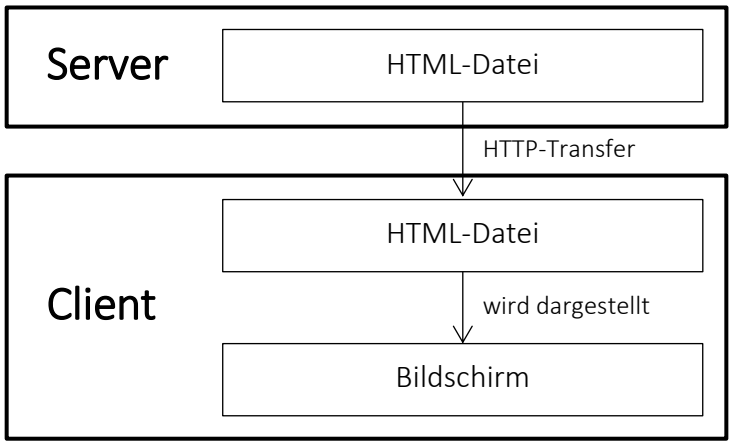

Abbildung 10: Statische Webseiten

Bei *statischen Webseiten* wird zwischen zwei Arten differenziert:

3.1.1 Manuell aktualisierte statische Webseiten

Bei *manuell aktualisierten statischen Webseiten* liegen die Dateien auf dem *Server* und werden von der sie anbietenden Person manuell aktualisiert.

Ein einfaches Beispiel hierfür ist die persönliche *Webseite* einer Person, oftmals als *Homepage* bezeichnet. Persönliche *Homepages* zielen meist darauf ab, die sie betreibende Person zu präsentieren. Die Beschreibung von Hobbys, persönlichen Vorlieben, sowie persönliche Daten, wie der eigene Geburtstag oder die E-Mail-Adresse, aber auch Neuigkeiten aus dem Leben der Person sind Beispiele für Inhalte dieser Art. Bei besagten Inhalten muss die die Webseite anbietende Person selbst über die Aktualisierung entscheiden.

3.1.2 Automatisch aktualisierte statische Webseiten

Jenes Merkmal, das die Art der *automatisch aktualisierten statischen Webseiten* auszeichnet, ist, dass die statischen Dateien automatisch aktualisiert werden.

Es gibt neben den obig bereits angesprochenen, über einen längeren Zeitraum gültigen Inhalten auch solche, die nur über einen kürzeren Zeitraum hinweg gültig sind. Das heutige, durch Tag, Monat und Jahr angegebene Datum ist ein Beispiel für einen derartigen Inhalt. Es ist in der Regel für 24 Stunden gültig. Ein weiteres Beispiel für Inhalte dieser Art ist die Wartezeit auf der Anzeigetafel einer Wiener Busstation, deren Daten nur eine Gültigkeitsdauer im Minutenbereich aufweisen.

Eine Möglichkeit, derartige Inhalte in eine *Webseite* einzuspeisen, besteht darin, die sie beinhaltenden Dateien in passenden Zeitabständen automatisch zu aktualisieren.

Dies kann im Fall des Datums beispielsweise durch ein Programm geschehen, welches die das Datum beinhaltende Datei automatisch aktualisiert und diese um 00:00 Uhr jede Nacht auf den *Server* hochlädt oder diese direkt auf dem *Server* manipuliert.

Die Wahl des Aktualisierungszeitpunkts hängt also vom Inhalt ab.

3.2 Dynamische Webseiten

Dynamische Webseiten hingegen zeichnen sich dadurch aus, dass der *Client* nicht exakt die gleiche Datei erhält, die auf dem *Server* gespeichert ist, sondern eine Datei, welche erst beim Abruf aus jenen Anweisungen, die in der abgerufenen Datei festgelegt sind, generiert wird.

Hierbei wird zwischen zwei unterschiedlichen Paradigmen differenziert: der *server*seitigen sowie der *client*seitigen Generierung. Diese werden im Folgenden beschrieben.

Festzuhalten ist, dass die beiden angesprochenen Paradigmen einander nicht gegenseitig ausschließen. Eine *serverseitig generierte dynamische Webseite* kann *client*seitig generierte Elemente enthalten und umgekehrt.

3.2.1 Serverseitig generierte dynamische Webseiten

Serverseitig generierte dynamische Webseiten basieren zumeist[54] auf der Programmiersprache *PHP Hypertext Processor* (*PHP*)[55], oder auch auf *Active Server Pages .NET* (*ASP.NET*)[56] oder *JavaServer Pages* (*JSP*)[57]. Der grundsätzliche Ablauf des Abrufs einer *serverseitig generierten Webseite* lässt sich wie folgt beschreiben:

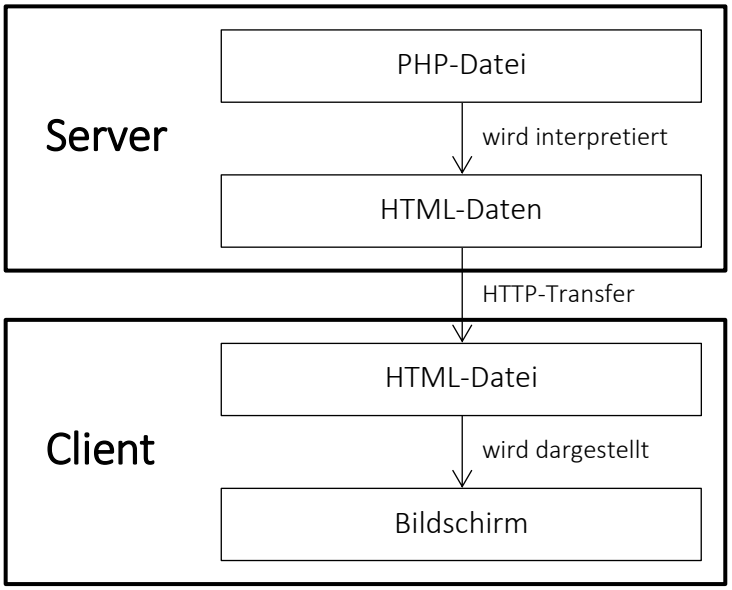

Abbildung 11: Serverseitig generierte dynamische Webseiten

Auf dem *Server* befindet sich eine Datei, die mit Instruktionen versehen ist, denen zufolge beim Abruf eine Datei generiert und an den *Client* geliefert wird. Ein Beispiel hierfür ist die bereits in Kapitel 3.1 angesprochene Ausgabe des heutigen Datums. Bei *serverseitig generierten dynamischen Webseiten* steht in der abzurufenden Datei ein *Befehl*, der veranlasst, dass vom *Server* oder einer anderen Quelle das heutige Datum abgeholt und dann beim *Client* abgeliefert wird.

[54] vgl. http://redmonk.com/sogrady/2013/07/25/language-rankings-6-13/ (19.01.2016)
[55] vgl. http://php.net/ (19.01.2016)
[56] vgl. http://www.asp.net/ (19.01.2016)
[57] vgl. http://www.oracle.com/technetwork/java/javaee/jsp/index.html (19.01.2016)

Der *Quelltext* bleibt für den *Client* im Verborgenen, da er auf dem *Server* ausgeführt wird und lediglich die Ausgabe dieser Ausführung an den *Client* geschickt wird.[58]

Anzumerken ist, dass die Interpretation von im *Quelltext* enthaltenen *Befehlen browser*unabhängig vonstattengeht, da sie von der auf dem *Server* installierten Software abhängt.

3.2.2 Clientseitig generierte dynamische Webseiten

Clientseitig generierte dynamische Webseiten basieren zumeist[59] auf der Sprache *JavaScript*[60]. Der grundsätzliche Ablauf bei der *client*seitigen Generierung lässt sich wie im Folgenden dargelegt beschreiben:

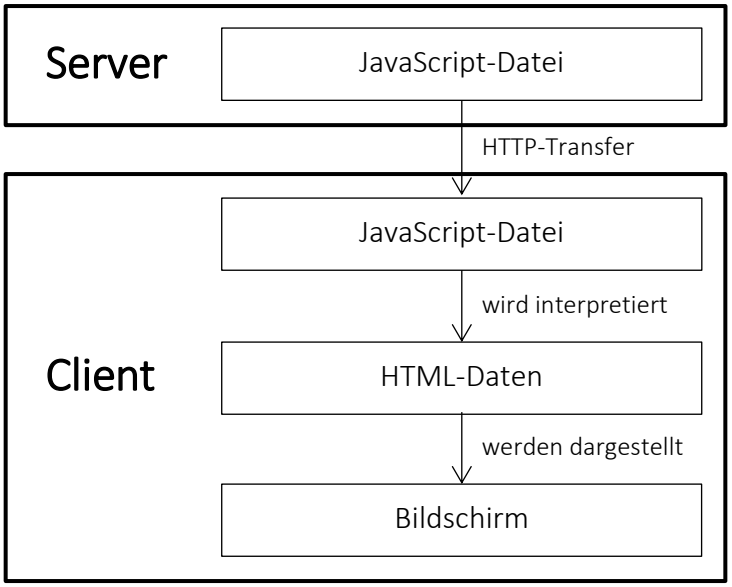

Abbildung 12: Clientseitig generierte dynamische Webseiten

Auf dem *Server* befindet sich eine Datei, die steuert, wie die *Webseite* schlussendlich aussieht. Zum Beispiel kann der *Server* dem *Client* mitteilen, dass er eine Überschrift erstellen soll, die einen bestimmten Text oder auch einen bestimmten Button beinhaltet. Die Generierung wird hierbei zwar durch den *Server* gesteuert, sie erfolgt aber erst beim *Client*. Der Client erhält also keine fertige Webseite, sie wird erst von diesem gefertigt. Hierfür muss dem *Client* der *Quelltext* bekannt sein, da der *Browser* und nicht der *Server* die Instanz ist, welche die darin enthaltenen *Befehle* ausführt. Der *Quelltext* kann von einer Person ausge-

[58] vgl. http://php.net/manual/de/intro-whatis.php (19.01.2016)
[59] vgl. http://redmonk.com/sogrady/2013/07/25/language-rankings-6-13/ (19.01.2016)
[60] vgl. http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf (19.01.2016) und https://developer.mozilla.org/en-US/docs/Web/JavaScript (19.01.2016)

lesen werden, da die ausführende Instanz von ihr selbst kontrolliert wird, was beim *Server* nicht der Fall ist.

Anzumerken ist, dass die Interpretation von *JavaScript-Befehlen browser*abhängig vonstattengeht, da manche *Browser* eine eigene Spezifikation von *JavaScript* verwenden.[61] Ob sich diese an die standardisierte *ECMA-International*-Spezifikation[62] hält, liegt im Ermessen der Entwicklerinnen und Entwickler des *Browsers*.

3.2.3 Hybride dynamische Webseiten

Hybride dynamische Webseiten sind eine Mischform aus *serverseitig generierten dynamischen Webseiten* und *clientseitig generierten dynamischen Webseiten*. Ein beispielhafter Ablauf einer hybriden Generierung lässt sich wie folgt darstellen:

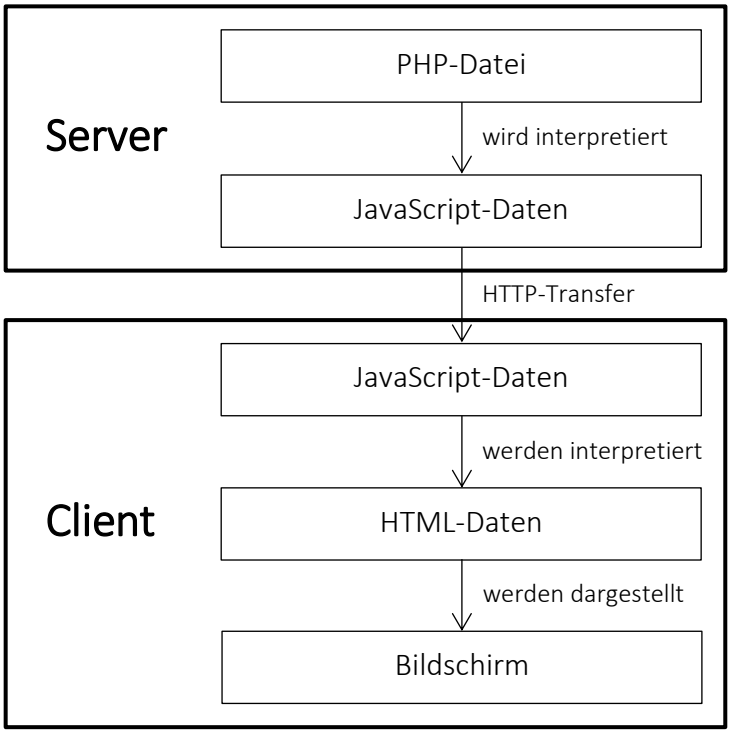

Abbildung 13: Hybride dynamische Webseiten

Hybride dynamische Webseiten sind *Webseiten*, die sowohl Elemente einer *serverseitig generierten dynamischen Webseite* als auch welche einer *clientseitig generierten dynamischen Webseite* enthalten. Die Generierung wird auch hierbei durch den *Server* gesteuert. Sie

[61] vgl. https://developer.mozilla.org/en-US/docs/Web/JavaScript (19.01.2016)
[62] vgl. http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf (19.01.2016)

erfolgt teilweise auf dem *Server*, teilweise beim *Client*. Die Zugänglichkeit des *Quelltexts* ist elementabhängig.

3.2.4 Serverseitige Generierung versus clientseitige Generierung

Serverseitig generierte dynamische Webseiten zeichnen sich zusätzlich zu den bereits angesprochenen Merkmalen dadurch aus, dass die Rechenlast hauptsächlich beim *Server* liegt. Je komplexer die auszuführenden Operationen sind, desto ausgelasteter und folglich desto weniger performant ist der *Server*.

Bezüglich der *clientseitig generierten dynamischen Webseiten* ist hingegen zu bemerken, dass die Rechenlast großteils beim *Client* liegt. Bei schwächeren Computern wie zum Beispiel Smartphones kann dies zu starken Einbußen in Leistung und vor allem im Komfort der Bedienung führen.

3.2.5 Login-Bereiche

Ergänzend ist zu bemerken, dass geschützte Bereiche einer *Webseite*, sogenannte *Login-Bereiche*, eine *server*seitige Verarbeitung erfordern.

Beispiele für *Webseiten* mit *Login-Bereichen* sind elischa.brg19.at, Twitter, Facebook, YouTube und etliche mehr.

Das *Login* erfolgt dabei oftmals durch die Eingabe eines Account-Namens und des dazugehörigen Passworts.

Ein solcher *Login-Bereich* stellt besondere Anforderungen an ein Automatisierungs*tool*.

3.3 Relevanz der Art einer Webseite im Kontext dieser Arbeit

Bei Wahl eines geeigneten *Tools* muss, neben den obig genannten besonderen Anforderungen, vor allem auf die Art einer *Webseite* Rücksicht genommen werden.

Die im weiteren Verlauf der Arbeit genauer besprochenen *Tools* haben jeweils spezifische Anwendungsgebiete und eignen sich daher unterschiedlich gut für die verschiedenen Arten von *Webseiten*.

Für die Anwenderin beziehungsweise den Anwender ist es daher wichtig, bestmöglich über die *Webseite*, auf die besagte *Tools* angewendet werden sollen, Bescheid zu wissen, da dadurch eine möglichst genau auf die betreffende *Webseite* zugeschnittene Anwendung erfolgen kann.

4 Tools zum Abruf

Wie de facto alles in der Informatik, gehorcht auch die automatisierte Datenextraktion dem Grundprinzip „Eingabe → Verarbeitung → Ausgabe“, respektive einem linearen Ablauf. Darauf basierend lässt sich eine Einteilung in Kategorien vornehmen, in welcher „Tools zum Abruf“ die Eingabe repräsentiert, da durch den Abruf jene Daten gewonnen werden, welche in die für die Verarbeitung zuständigen Programme eingespeist werden. Anzumerken ist, dass die Klassifizierung hinsichtlich der Grundfunktionalität der einzelnen *Tools* getroffen worden ist.

4.1 `wget`

Das erste besprochene *Tool* zum Abruf ist `wget`[63]. Dieses ist ein unter der *GPL-Lizenz*[64] veröffentlichtes Programm zum Download von Informationen aus dem Internet. Es unterstützt die *Protokolle HTTP*, *HTTPS* und *FTP* und ist in der Programmiersprache *C*[65] geschrieben.

Bei `wget` handelt es sich um ein nicht-interaktives Programm. Daher kann es leicht in *Skripten* oder *Cronjobs* aufgerufen werden. *Cronjobs* sind automatisiert zeitgesteuert ausgeführte wiederkehrende Aufgaben.[66] Neben dem Download einzelner Dateien unterstützt `wget` auch das Speichern ganzer *Webseiten* mitsamt den Bildern, beispielsweise zum Zweck des Offline-Lesens.[67] Es bietet ferner die Möglichkeit der Wiederaufnahme von unterbrochenen Downloads. `wget` eignet sich erfahrungsgemäß ausgesprochen gut zum automatisierten Download von *Webseiten*.

Bei `wget` werden die Daten, die übertragen werden sollen, in Form von *URLs* angegeben.

Die allgemeine *Syntax* sieht wie folgt aus:[68]

```
wget [Option] …  [URL] …
```

[63] vgl. https://www.gnu.org/software/wget/ (19.01.2016)
[64] vgl. http://www.gnu.org/licenses/gpl-3.0.de.html (19.01.2016)
[65] vgl. u. a. http://csapp.cs.cmu.edu/3e/docs/chistory.html (19.01.2016)
[66] vgl. u. a. http://pubs.opengroup.org/onlinepubs/9699919799/utilities/crontab.html (19.01.2016)
[67] vgl. https://www.gnu.org/software/wget/manual/wget.html#Overview (19.01.2016)
[68] vgl. https://www.gnu.org/software/wget/manual/wget.html#Invoking (19.01.2016)

4.1.1 Optionen von `wget`

`wget` bietet der Benutzerin beziehungsweise dem Benutzer verschiedene *Optionen*. Wichtige davon werden im Folgenden erläutert.

Option	Beschreibung
`-i` *datei* `--input-file=`*datei*	Diese *Option* veranlasst, dass die *URLs* der Dateien, die übertragen werden sollen, aus der angegebenen Textdatei namens *datei* gelesen werden.[69]
`-b` `--background`	Diese *Option* bewirkt, dass der Download als Hintergrundprozess durchgeführt wird. Wenn nicht anders angegeben, werden alle Statusmeldungen in die Datei `wget-log` geschrieben, die im *Arbeitsverzeichnis* erstellt wird.[70]
`-B` *url* `--base=`*url*	Diese *Option* hat zur Folge, dass allen relativen *Links* innerhalb der Datei, die die Benutzerin beziehungsweise der Benutzer durch `-i` angegeben hat, die Basis*adresse url* vorangestellt wird.[71]
`-c` `--continue`	Diese *Option* veranlasst, dass ein unterbrochener Download wiederaufgenommen wird. Wenn sich die Datei allerdings in der Zwischenzeit auf dem *Server* geändert hat, ist die heruntergeladene Datei eventuell fehlerhaft.[72]
`-F` `--force-html`	Diese *Option* bewirkt, dass die über die *Option* `-i` angegebene Datei als Datei im *HTML*-Format interpretiert wird, und dass alle Dateien, auf welche *Links* der *HTML*-Datei verweisen, geladen werden.[73]
`--limit-rate=`*n*	Durch diese *Option* wird die Download-Menge pro Sekunde auf die durch *n* angegebene Bytezahl limitiert. Die Suffixe `k` und `m` bezeichnen hierbei Kilo- beziehungsweise Megabytes.[74]

[69] vgl. https://www.gnu.org/software/wget/manual/wget.html#Logging-and-Input-File-Options (19.01.2016)
[70] vgl. https://www.gnu.org/software/wget/manual/wget.html#Basic-Startup-Options (19.01.2016)
[71] vgl. https://www.gnu.org/software/wget/manual/wget.html#Logging-and-Input-File-Options (19.01.2016)
[72] vgl. https://www.gnu.org/software/wget/manual/wget.html#Download-Options (19.01.2016)
[73] vgl. https://www.gnu.org/software/wget/manual/wget.html#Logging-and-Input-File-Options (19.01.2016)
[74] vgl. https://www.gnu.org/software/wget/manual/wget.html#Download-Options (19.01.2016)

`-N`	Diese *Option* veranlasst, dass eine Datei nur dann heruntergeladen wird, wenn diese neuer ist als eine gleichnamige Datei, die bereits auf dem Computer gespeichert ist.[75]
`--output-document=`*`datei`*	Durch diese *Option* wird der heruntergeladenen Datei der Name *`datei`* gegeben.
`-P` *`verzeichnis`*	Diese *Option* sorgt dafür, dass alle heruntergeladenen Dateien im *Verzeichnis* *`verzeichnis`* gespeichert werden.[76]
`-q` `--quiet`	Ist diese *Option* aktiviert, werden keine Statusmeldungen ausgegeben.[77]
Optionen für rekursives Herunterladen	
`-E` `--html-extension`	Diese *Option* sorgt dafür, dass die Dateienendung `.html` bei allen heruntergeladenen Dateien des *MIME-Types* `application/xhtml+xml` oder `text/html` angefügt wird, sofern deren Name nicht mit `.html` beziehungsweise `.htm` endet.[78]
`-H` `--span-hosts`	Durch diese *Option* werden auch *Links*, die auf andere *Webseiten* verweisen, verfolgt.[79]
`-k` `--convert-links`	Durch diese *Option* werden bei den heruntergeladenen *HTML*-Dateien die externen *Links* so geändert, dass sie auf die lokalen Dateien verweisen.[80]
`-r` `--recursive`	Diese *Option* veranlasst, dass *rekursiv heruntergeladen* wird.[81]
`-p` `--page-requisites`	Durch diese *Option* werden alle Dateien heruntergeladen, welche zum Betrachten der Ausgangsdatei erforderlich sind.[82]

Tabelle 3: Optionen von `wget`

[75] vgl. https://www.gnu.org/software/wget/manual/wget.html#Download-Options (19.01.2016)
[76] vgl. https://www.gnu.org/software/wget/manual/wget.html#Directory-Options (19.01.2016)
[77] vgl. https://www.gnu.org/software/wget/manual/wget.html#Logging-and-Input-File-Options (19.01.2016)
[78] vgl. https://www.gnu.org/software/wget/manual/wget.html#HTTP-Options (19.01.2016)
[79] vgl. https://www.gnu.org/software/wget/manual/wget.html#Recursive-Accept_002fReject-Options (19.02.2016)
[80] vgl. https://www.gnu.org/software/wget/manual/wget.html#Recursive-Retrieval-Options (19.01.2016)
[81] vgl. https://www.gnu.org/software/wget/manual/wget.html#Recursive-Retrieval-Options (19.01.2016)
[82] vgl. https://www.gnu.org/software/wget/manual/wget.html#Recursive-Retrieval-Options (19.02.2016)

4.1.2 Beispieleingaben zu `wget`

Im Folgenden werden einige Beispieleingaben zur Verwendung von `wget` erläutert.

Beispiel 1: `wget http://www.brg19.at/public.php/2/86`

Diese Eingabe lädt die unter der *Adresse* `http://www.brg19.at/public.php/2/86` befindliche *Webseite* herunter. Am 22. Jänner 2016 war dies das Schulprofil des BRG19.

Beispiel 2: `wget -c http://www.brg19.at/public.php/2/86`

Wurde der Download aus irgendeinem Grund abgebrochen, kann er über das Aktivieren der *Option* `-c` wiederaufgenommen werden.

Beispiel 3: `wget -b http://www.brg19.at/public.php/2/86`

Diese *Option* lässt den Download im Hintergrund ablaufen. Das bietet sich vor allem bei großen Dateien an, die mehr Downloadzeit in Anspruch nehmen.

Beispiel 4: `wget -k http://www.brg19.at/public.php/2/86`

Alle externen *Links* werden so verändert, dass sie auf lokale Dateien verweisen. Dies bietet sich an, wenn man eine *Webseite* offline lesen möchte.

4.1.3 Anwendung von `wget` auf die verschiedenen Arten von Webseiten

`wget` kann zur Gänze auf die Art der statischen *Webseiten* angewendet werden. Schwierigkeiten treten jedoch unter anderem bei *Webseiten*, welche mit einem *Login-Bereich* geschützt sind, auf, da `wget` diese nicht downloaden kann.

4.2 `curl`

`curl`[83] ist ein weiteres Programm zum Download von *Webseiten*. Es ist ein *Open Source Kommandozeilenprogramm*, mit dem man Dateien in der *URL-Syntax* von und zu einem *Server* übertragen kann. Es unterstützt eine Vielzahl an *Protokollen*, unter anderem auch das *HTTP*.[84] Auch `curl` kann in *Skripten* aufgerufen werden.

Die allgemeine *Syntax* sieht wie folgt aus:[85]

```
curl [Option] …  [URL] …
```

[83] vgl. http://curl.haxx.se/ (19.01.2016)
[84] vgl. http://curl.haxx.se/, curl is an open source command line tool and library (19.01.2016)
[85] vgl. http://curl.haxx.se/docs/manpage.html, SYNOPSIS (19.01.2016)

4.2.1 Optionen von `curl`

`curl` bietet der Benutzerin beziehungsweise dem Benutzer verschiedene *Optionen*. Die wichtigsten werden im Folgenden erläutert.[86]

Option	Beschreibung
`--limit-rate` *n*	Durch diese *Option* wird die Download-Menge pro Sekunde auf die durch *n* angegebene Bytezahl limitiert. Die Suffixe `k` und `m` bezeichnen hierbei Kilo- beziehungsweise Megabytes.
`-o` *datei*	Diese *Option* veranlasst, dass die heruntergeladenen Daten in einer Datei namens *datei* gespeichert werden, anstatt sie über die *Standardausgabe* auszugeben.
`-r` *n1-n2*	Durch diese *Option* wird der durch *n1* und *n2* angegebene Bytebereich der Datei übertragen.

Tabelle 4: Optionen von `curl`

4.2.2 Beispieleingaben zu `curl`

Im Folgenden werden einige Beispieleingaben zur Verwendung von `curl` erläutert.

Beispiel 1: `curl http://www.brg19.at/public.php/2/86`
Diese Eingabe lädt die unter der *Adresse* `http://www.brg19.at/public.php/2/86` befindliche *Webseite* herunter. Am 22. Jänner 2016 war dies das Schulprofil des BRG19.

Beispiel 2: `curl -r 2012-2016 http://www.brg19.at/public.php/27/2`
Diese Eingabe lädt den Inhalt der Datei, der sich im Bytebereich 2012 bis 2016 befindet, herunter. Am 22. Jänner 2016 war das das Wort „Stand".

4.2.3 Anwendung von `curl` auf die verschiedenen Arten von Webseiten

Die Anwendung auf die verschiedenen Arten von *Webseiten* betreffend gilt für `curl` dasselbe wie für `wget`.

`curl` ist ebenso wie `wget` ein eigenständiges *Tool*, das ohne einen *Browser* auskommt.

[86] vgl. http://curl.haxx.se/docs/manpage.html, OPTIONS (19.01.2016)

4.3 Selenium

Selenium hingegen kommt nicht ohne einen *Browser* aus. Es ist nämlich eine Ansammlung von *Tools*, die sich zum Ziel gesetzt hat, den Test von webbasierten Anwendungen zu automatisieren, es ist folglich eine Automatisierung von *Browsern*.[87]

4.3.1 Selenium IDE

In dieser Arbeit wird lediglich ein *Tool* aus der erwähnten Ansammlung genauer besprochen, *Selenium IDE*. Dieses ist ein *Addon* für *Mozilla Firefox*. Es ermöglicht, einzelne Abläufe von Aktionen im *Browser* aufzunehmen und diese später wiederholt abzuspielen. Außerdem bietet es die Möglichkeit, diese Aufnahme in der *Syntax* diverser Programmiersprachen zu exportieren und somit die Automatisierung in *Skripten* zu ermöglichen.[88]

Allerdings unterstützt *Selenium IDE* einige *Befehle* aus anderen *Selenium-Tools* nicht.[89] Hierbei kann man sich Abhilfe schaffen, indem man nach dem Export die *Skripte* dahingehend editiert, dass die fehlenden Funktionen entsprechend von Hand ergänzt werden.

Die Benutzeroberfläche von *Selenium IDE* sieht wie folgt aus:

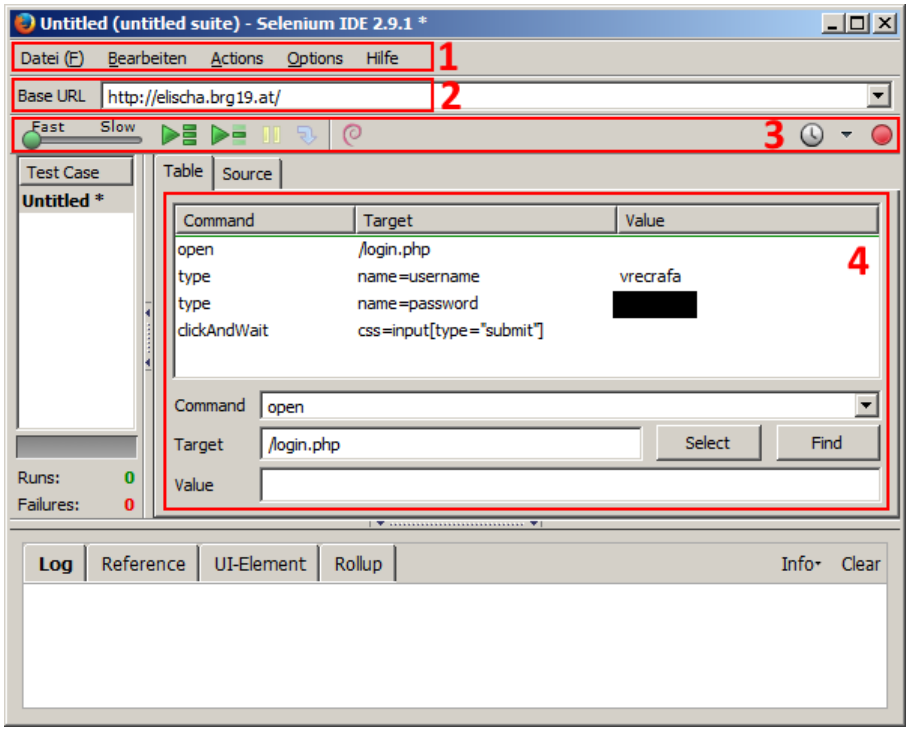

Abbildung 14: Benutzeroberfläche von Selenium IDE

[87] vgl. http://docs.seleniumhq.org/ (19.01.2016)
[88] vgl. http://docs.seleniumhq.org/docs/02_selenium_ide.jsp (19.01.2016)
[89] vgl. http://www.seleniumhq.org/docs/ (19.01.2016)

1 Menu Bar: Diese beinhaltet diverse Programm*optionen*. Im Menüpunkt „Datei" wird unter anderem die bereits erwähnte *Option* angeboten, Aufnahmen in der *Syntax* verschiedener Programmiersprachen zu exportieren.[90]
2 Base *URL*: Diese ist die *URL*, von welcher ausgehend mit der Ausführung der *Befehlskette* begonnen werden soll.
3 *Tool*bar: Die *Tool*bar bietet verschiedene *Optionen* an. Man kann eine vollständige Aufnahme oder auch einzelne Teile davon abspielen. Ebenfalls wichtig ist der Record-Button auf der rechten Seite, der besagte Aufnahmen ermöglicht.[91]
4 Test Case Pane: In diesem Fenster werden die einzelnen *Befehle* aufgelistet. Es können weitere von Hand ergänzt beziehungsweise bestehende bearbeitet werden.[92]

In diesem Fall ist in Abbildung 14 bereits das *Einloggen* des Users „vrecrafa" in das Portal elischa.brg19.at aufgezeichnet worden.

4.3.2 Befehle von Selenium

Folgende Tabelle beinhaltet drei wichtige *Selenium-Befehle*:

Befehl	Beschreibung
`open (URL)`	Durch Eingabe dieses *Befehls* wird die *URL* `URL` geöffnet.
`type (locator,value)`	Dieser *Befehl* ermöglicht die Eingabe eines Wertes (`value`) in ein Feld (`locator`).
`clickAndWait`	Durch Eingabe dieses *Befehls* wird *Selenium* mitgeteilt, dass nach dem Klicken auf ein Element (beispielsweise ein Button) eine Interaktion mit dem *Server* stattfindet.

Tabelle 5: Befehle von Selenium (angewendet in Selenium IDE)[93]

4.3.3 Anwendung von Selenium auf die verschiedenen Arten von Webseiten

Selenium ist das mächtigste der hier besprochenen *Tools* zum Abruf. Es ist de facto uneingeschränkt auf die verschiedenen Arten von *Webseiten* anwendbar. Da es *Browser* automatisiert, stellen für *Selenium* auch *Login-Bereiche* kein Problem dar, mit *JavaScript* generierte *Webseiten* ebenso wenig.

[90] vgl. http://docs.seleniumhq.org/docs/02_selenium_ide.jsp, IDE Features, Menu Bar (19.01.2016)
[91] vgl. http://docs.seleniumhq.org/docs/02_selenium_ide.jsp, IDE Features, Toolbar (19.01.2016)
[92] vgl. http://docs.seleniumhq.org/docs/02_selenium_ide.jsp, IDE Features, Test Case Pane (19.01.2016)
[93] vgl. http://release.seleniumhq.org/selenium-core/1.0.1/reference.html (19.01.2016)

5 Tools zur Verarbeitung

Nach dem Download der jeweiligen ***Webseite*** liegt der Inhalt, abhängig vom verwendeten ***Tool***, im ***HTML***-Format vor. Ziel der Weiterverarbeitung ist es, aus diesem die gewünschten Daten zu extrahieren. Zur Weiterverarbeitung sind ebenfalls ***Tools*** notwendig, welche im Folgenden besprochen werden. Die folgenden ***Tools*** sind allesamt in ***Skripten*** aufrufbar und eignen sich somit sehr gut für automatisierte Anwendungen.

5.1 GNU Coreutils und ähnliche

In diesem Zusammenhang ist das für diese Arbeit wichtigste Paket ***GNU Coreutils***[94]. Dieses ist eine Ansammlung von diversen Programmen zur nicht-interaktiven Textverarbeitung. In dieser Arbeit werden einzelne Programme aus diesem Paket inklusive ihrer ***Eingabeparameter*** besprochen. Einige Programme, die in ihrer Anwendung ähnlich sind, jedoch nicht zu diesem Paket gehören, werden im Folgenden ebenfalls besprochen.

5.1.1 `cat`[95]

`cat` gibt den Inhalt einer Textdatei aus. Es bietet außerdem die Möglichkeit, mehrere einzelne Dateien zu einer größeren Datei zusammenzufügen.

Die allgemeine ***Syntax*** lautet wie folgt:

```
cat [Option] …  [Datei] …
```

`cat` verfügt über mehrere ***Optionen***, einige wichtige davon sind:

Option	Beschreibung
`-s`	Durch diese ***Option*** werden mehrere Leerzeilen auf eine einzige Leerzeile reduziert.
`-T`	Diese ***Option*** veranlasst, dass ***Tabulatoren*** in der Form `^I` angezeigt werden.
`-v`	Diese ***Option*** veranlasst, dass Zeichen, die nicht druckbar sind, in einer speziellen Schreibweise angezeigt werden.

Tabelle 6: Optionen von `cat`

[94] vgl. http://www.gnu.org/software/coreutils/coreutils.html (22.01.2016)
[95] vgl. https://www.gnu.org/software/coreutils/manual/coreutils.html#cat-invocation (19.01.2016)

Beispiel: `cat testdatei.txt`

Diese Eingabe gibt den Inhalt der Textdatei `testdatei.txt` auf dem Bildschirm aus.

Die Datei `testdatei.txt` wurde zur Veranschaulichung der Funktionalität der einzelnen *Tools* erstellt. Ihr Inhalt und damit die Ausgabe des obig angeführten *Befehls* sieht wie folgt aus:

```
1 Vorwissenschaftliche Arbeit 2017
2 Vorwissenschaftliche Arbeit 2015
3 Vorwissenschaftliche Arbeit 2018
4 Vorwissenschaftliche Arbeit 2016
5 Wissenschaftliche Diplomarbeit 2020
6 Matura 2015
7 Volksschulabschluss 2000
8 Referat am 22. Oktober 2015
9 Vorstellung der neuen 1. Klasse um 13:00 Uhr
```

Quelltext 2: Ausgabe von `cat`

Die obig abgebildete Datei dient im weiteren Verlauf als Muster, auf welches die einzelnen *Tools* angewendet werden. Sie wird nicht für jedes Beispiel erneut abgebildet.

5.1.2 `cut`[96]

`cut` extrahiert zeilenweise aus einem Text jene Spalten, die durch die jeweiligen *Optionen* angegeben werden.

Die allgemeine *Syntax* des Programms lautet:

```
cut [Option] …  [Datei] …
```

`cut` weist verschiedene *Optionen* auf, folgende ist wichtig:

Option	Beschreibung
`-c` *`liste`* `--characters=`*`liste`*	Bei dieser *Option* werden Zeichen abgezählt. Der *Parameter* *`liste`* gibt an, welche und wie viele Zeichen ausgegeben werden.

Tabelle 7: Eine wichtige Option von `cut`

Beispiel: `cut -c 1-19 testdatei.txt`

Diese Eingabe gibt die Zeichen Nummer 1 bis 19 jeder Zeile der angeführten Datei aus.

[96] vgl. https://www.gnu.org/software/coreutils/manual/coreutils.html#cut-invocation (19.01.2016)

Die Ausgabe sieht so aus:

```
                1
1234567890123456789
```

1	Vorwissenschaftlich
2	Vorwissenschaftlich
3	Vorwissenschaftlich
4	Vorwissenschaftlich
5	Wissenschaftliche D
6	Matura 2015
7	Volksschulabschluss
8	Referat am 22. Okto
9	Vorstellung der neu

Quelltext 3: Ausgabe von `cut`

5.1.3 grep[97]

`grep` durchsucht Dateien nach bestimmten Suchmustern und zeigt die Zeilen an, in welchen diese gefunden wurden. `grep` ist zwar ein *Tool* des GNU-Projekts, gehört aber nicht zum Paket der *GNU Coreutils*.

Die allgemeine *Syntax* lautet wie folgt:

```
grep [Option] …  Suchmuster [Datei] …
```

`grep` stellt eine Vielzahl an *Optionen* bereit, folgende werden besprochen:

Option	Beschreibung
`-n`	Diese *Option* veranlasst `grep` dazu, nicht nur die Zeile mit dem gefundenen Textstück anzuzeigen, sondern auch die ***n*** vorangehenden und nachfolgenden Zeilen.
`-i`	Mit dieser *Option* wird bei der Suche nicht zwischen Groß- und Kleinschreibung unterschieden.
`-l`	Diese *Option* gibt die Dateinamen der Dateien aus, in denen das eingegebene Suchmuster gefunden wurde.
`-v`	Diese *Option* hat eine inverse Wirkung. Das heißt, dass sie alle Zeilen ausgibt, die dem Suchmuster nicht entsprechen.

Tabelle 8: Optionen von `grep`

[97] vgl. http://www.gnu.org/software/grep/manual/grep.html (19.01.2016)

Beispiel: `grep wissenschaft testdatei.txt`

Diese Eingabe gibt diejenigen Zeilen aus, in welchen „wissenschaft“ gefunden wird. Das Wort wird farblich hervorgehoben und **fett** gedruckt. Die Groß- und Kleinschreibung wird hierbei berücksichtigt.

Die Ausgabe sieht also wie folgt aus:

```
1 Vorwissenschaftliche Arbeit 2017
2 Vorwissenschaftliche Arbeit 2015
3 Vorwissenschaftliche Arbeit 2018
4 Vorwissenschaftliche Arbeit 2016
```

Quelltext 4: Ausgabe von `grep`

5.1.4 `head`[98] und `tail`[99]

`head` beziehungsweise `tail` gibt die ersten beziehungsweise die letzten zehn Zeilen einer Datei auf dem Bildschirm aus, sofern kein *Parameter* übermittelt wird.

Über die *Option* `-n` *x* lässt sich die Zahl von zehn auf *x* Zeilen verändern.

Für `head` gilt außerdem: Falls dieser Zahl das Zeichen „-“ vorangestellt ist, werden nicht die ersten *x* Zeilen, sondern die letzten *x* Zeilen ausgegeben.

Für `tail` gilt hingegen: Falls dieser Zahl das Zeichen „+“ vorangestellt ist, werden alle Zeilen ab inklusive der angegebenen Zeile ausgegeben.

Die allgemeine *Syntax* lautet:

```
head/tail [Option] …  [Datei] …
```

Beispiel: `head -n -7 testdatei.txt`

Diese Eingabe gibt alle Zeilen von `testdatei.txt` außer die letzten sieben aus:

```
1 Vorwissenschaftliche Arbeit 2017
2 Vorwissenschaftliche Arbeit 2015
```

Quelltext 5: Ausgabe von `head`

[98] vgl. https://www.gnu.org/software/coreutils/manual/coreutils.html#head-invocation (19.01.2016)
[99] vgl. https://www.gnu.org/software/coreutils/manual/coreutils.html#tail-invocation (19.01.2016)

Die Ausgabe aller Zeilen außer diesen beiden, sprich die inverse Ausgabe zu der eben genannten, kann durch den *Befehl* `tail -n +3 testdatei.txt` erzielt werden. `tail` gibt dabei alle Zeilen ab inklusive der dritten aus.

5.1.5 `sed`

`sed` steht für Stream Editor.[100] Mit ihm können Text-*Datenströme* editiert werden. Der Begriff „*Datenstrom*" bezeichnet in der Informatik einen kontinuierlichen Fluss von Datensätzen.[101] `sed` verarbeitet Datensätze fortlaufend, sobald jeweils ein neuer Datensatz eingetroffen ist.[102] `sed` ist zwar ein *Tool* des GNU-Projekts, gehört aber nicht zum Paket der *GNU Coreutils*.

Die allgemeine *Syntax* von `sed` sieht so aus:[103]

```
sed [Option] …  [Script] [Datei] …
```

Da `sed` in der Anwendung wesentlich komplizierter als die bereits besprochenen *Tools* ist, wurde die Erklärung auf jene Grundlagen beschränkt, die für die automatisierte Datenextraktion im Kontext dieser Arbeit erforderlich sind.

Die *Eingabeparameter* werden bei `sed` durch sogenannte *Delimiter* (Trennzeichen) getrennt, welche für die Anwendung passend zu wählen sind.

Beispiel: `sed 's/Vorwissenschaftliche/Wissenschaftliche/g' testdatei.txt`
Durch diese Eingabe werden alle Vorkommnisse von „Vorwissenschaftliche" in „Wissenschaftliche" geändert. Zu berücksichtigen ist, dass beispielsweise „Vorwissenschaftliches" in „Wissenschaftliches" geändert werden würde. Dabei ist irrelevant, wie viele Zeichen auf „Vorwissenschaftliche" folgen. Sie sind von der Änderung nicht betroffen. Die Groß- und Kleinschreibung wird hierbei berücksichtigt, also wäre „vorwissenschaftliche" nicht betroffen.

`s` steht hierbei für „substitute" (ersetze) und ersetzt Zeichen(*ketten*) durch andere. `g` steht hierbei für „global", was bedeutet, dass die Eingabe nicht nur auf das erste Vorkommnis

[100] vgl. https://www.gnu.org/software/sed/ (19.01.2016)
[101] vgl. http://www.its.bldrdoc.gov/fs-1037/dir-010/_1451.htm (19.01.2016)
[102] vgl. https://www.gnu.org/software/sed/manual/sed.html#Introduction (19.01.2016)
[103] vgl. https://www.gnu.org/software/sed/manual/sed.html#Invoking-sed (19.01.2016)

innerhalb jedes Datensatzes, sondern auf alle Vorkommnisse innerhalb jedes Datensatzes angewendet werden soll.[104]

Die Ausgabe sieht daher wie folgt aus:

```
10 Wissenschaftliche Arbeit 2017
11 Wissenschaftliche Arbeit 2015
12 Wissenschaftliche Arbeit 2018
13 Wissenschaftliche Arbeit 2016
14 Wissenschaftliche Diplomarbeit 2020
15 Matura 2015
16 Volksschulabschluss 2000
17 Referat am 22. Oktober 2015
18 Vorstellung der neuen 1. Klasse um 13:00 Uhr
```

Quelltext 6: Ausgabe von `sed`

Bei der obig angesprochenen Eingabe sind die „/"-Zeichen die ***Delimiter***. Ein Beispiel für eine Eingabe, in welcher die „/"-Zeichen nicht als ***Delimiter*** fungieren können, ist, wenn man exemplarisch in einem ***HTML***-Dokument alle schließenden ***HTML-Tags*** in eine andere ***Zeichenkette*** ändern will. Ein schließender ***HTML-Tag***, hier am Beispiel einer Überschrift erster Ebene, sieht wie folgt aus:

```
</h1>
```

Die Verwendung der „/"-Zeichen als ***Delimiter*** ist in diesem Fall deshalb nicht möglich, da die Eingabe ***syntaktisch*** falsch wäre, da sie vier als ***Delimiter*** zu erkennende Zeichen beinhaltet. Ein konkretes Beispiel:

```
sed 's/</h1>/VwA/g'
```

In einem solchen Fall muss ein anderes Zeichen als ***Delimiter*** verwendet werden. Eine korrekte Eingabe wäre daher:

```
sed 's#</h1>#VwA#g'
```

5.1.6 `sort`[105]

`sort` sortiert angegebene Dateien zeilenweise und gibt das Ergebnis davon auf dem Bildschirm aus.

[104] vgl. https://www.gnu.org/software/sed/manual/sed.html#index-Global-substitution-108 (19.01.2016)
[105] vgl. https://www.gnu.org/software/coreutils/manual/coreutils.html#sort-invocation (19.01.2016)

Die allgemeine *Syntax* von `sort` lautet:

```
sort [Option] …  [Datei] …
```

Einige wichtige *Optionen* von `sort` sind:

Option	Beschreibung
`-c`	Durch diese *Option* wird geprüft, ob die Datei bereits sortiert ist oder nicht.
`-f`	Durch diese *Option* werden Groß- und Kleinbuchstaben gleichwertig behandelt.
`-k` *n1,n2*	Durch diese *Option* werden nur die Zeichen zwischen der *n1*-ten und der *n2*-ten Spalte berücksichtigt. Die Spaltennummerierung beginnt hierbei mit „1“. Spalten werden in der Regel durch Leerzeichen oder *Tabulatoren* getrennt.
`-r`	Durch diese *Option* wird in umgekehrter Reihenfolge sortiert.
`-tz`	Mit dieser *Option* wird das Trennzeichen zwischen zwei Spalten angegeben.

Tabelle 9: Optionen von `sort`

Beispiel: `sort testdatei.txt`

Diese Eingabe sortiert die Datei `testdatei.txt` zeilenweise, beginnend mit dem ersten Zeichen der jeweiligen Zeile.

Die Ausgabe sieht wie folgt aus:

```
1 Matura 2015
2 Referat am 22. Oktober 2015
3 Volksschulabschluss 2000
4 Vorstellung der neuen 1. Klasse um 13:00 Uhr
5 Vorwissenschaftliche Arbeit 2015
6 Vorwissenschaftliche Arbeit 2016
7 Vorwissenschaftliche Arbeit 2017
8 Vorwissenschaftliche Arbeit 2018
9 Wissenschaftliche Diplomarbeit 2020
```

Quelltext 7: Ausgabe von `sort`

5.1.7 `tr`[106]

`tr` ersetzt oder löscht Zeichen einer Eingabe und gibt das Ergebnis über die ***Standardausgabe*** aus.

Die allgemeine ***Syntax*** von `tr` lautet:

```
tr [Option] …  Zeichenfolge1 [Zeichenfolge2]
```

Eine wichtige ***Option*** von `tr` ist `-d`. Diese löscht die angegebenen Zeichen. In diesem Fall muss keine zweite ***Zeichenfolge*** mitgeliefert werden.

Beispiel: `tr 'V' 'U' < testdatei.txt`
Durch diese Eingabe wird jedes „V“ in der Datei `testdatei.txt` durch „U“ ersetzt. Die Groß- und Kleinschreibung wird hierbei berücksichtigt. Bei `tr` kann die Eingabedatei nicht als ***Parameter*** mitgeliefert werden. Das Zeichen „<“ bedeutet ***befehls***unabhängig, dass die Eingabe aus der jeweiligen Datei bezogen werden soll.

Die Ausgabe sieht so aus:

```
1 Uorwissenschaftliche Arbeit 2017
2 Uorwissenschaftliche Arbeit 2015
3 Uorwissenschaftliche Arbeit 2018
4 Uorwissenschaftliche Arbeit 2016
5 Wissenschaftliche Diplomarbeit 2020
6 Matura 2015
7 Uolksschulabschluss 2000
8 Referat am 22. Oktober 2015
9 Uorstellung der neuen 1. Klasse um 13:00 Uhr
```

Quelltext 8: Ausgabe von `tr`

5.1.8 Abschlussbemerkungen zu den GNU Coreutils

Die ***GNU Coreutils*** sind ein sehr mächtiges ***Werkzeug*** im Umgang mit Text, mit welchem gerade durch die Automatisierung in ***Skripten*** viele Anwendungsfälle abgedeckt werden können, so auch die automatisierte Datenextraktion aus ***Webseiten***. Allerdings stoßen die ***GNU Coreutils*** dann an ihre Grenzen, wenn sich die Struktur einer ***Webseite*** ändert beziehungsweise Unregelmäßigkeiten in dieser auftreten. Diese Grenzen werden im Folgenden erläutert. Die ***Tools*** `grep` und `sed` gehören zwar nicht zum Paket der ***GNU Coreutils***, die hier getroffenen Bemerkungen gelten für sie trotzdem.

[106] vgl. https://www.gnu.org/software/coreutils/manual/coreutils.html#tr-invocation (19.01.2016)

5.2 Parser

Das in Kapitel 5.1.2 zu `cut` angeführte Beispiel ist ein Hinweis darauf, dass die *GNU Coreutils* gewisse Einschränkungen mit sich bringen.

Versucht man mit `cut` beispielsweise gewisse Zeichen jeder Zeile zu extrahieren, sind die erhaltenen Daten eventuell schon unbrauchbar, wenn nur ein Leerzeichen zu viel gesetzt wird, obwohl die überflüssigen Leerzeichen in *HTML* keine Änderung in der Darstellung bewirken.

Ein Beispiel hierfür ist das manuelle Eintragen von Daten über Online-Formulare. Angenommen, jede Lehrperson der Schule trägt ihre Wahlmodulnummer manuell in eine Tabelle ein, dann liefert `cut`, sofern aus Versehen nur ein Leerzeichen zwischen den Ziffern gesetzt wird, ein falsches Ergebnis zurück.

Eine komplexere Struktur kann beispielsweise auch die Sprechstundenliste einer Schule haben. Angenommen, die Sprechstunde einer Lehrperson entfällt jeden ersten Donnerstag im Monat aufgrund von Fortbildung, so ist die beispielhafte Struktur

```
Vorname Nachname
Sprechstunde
```

bereits unregelmäßig, da sie bei besagter Lehrperson wie folgt aussähe:

```
Vorname Nachname
Sprechstunde
Der erste Donnerstag jedes Monats entfällt aufgrund von Fortbildung!
```

Besagte Lehrperson benötigt drei Zeilen, während die restlichen Lehrpersonen lediglich zwei Zeilen benötigen. Will man nun beispielsweise die Sprechstunden herausfiltern und extrahiert mit einem Programm jede zweite Zeile, da in dieser üblicherweise die Sprechstunde steht, so erhält man ab der Zeile nach der Anmerkung die falschen Daten, da sich die Struktur verschoben hat und die Folgezeile keine Sprechstunde mehr ist, wie sie es eigentlich sein sollte, sondern durch die Verschiebung einen Vornamen und einen Nachnamen beinhaltet.

Aus *Webseiten*, die eine sehr komplexe Struktur haben, ist es also schwierig, mithilfe der *GNU Coreutils* zuverlässig Daten zu extrahieren. In diesem Fall kommen sogenannte *Parser* zur Anwendung, um Abhilfe zu schaffen.

Ein ***Parser*** ist ein Programm, das eine beliebige Eingabe zerlegt und in ein für die Weiterverarbeitung brauchbares Format umwandelt.

Allgemein wird ein ***Parser*** verwendet, um einen Text in eine neue Struktur zu übersetzen. Dabei müssen ***lexikalische*** und ***grammatische Analysen*** durchgeführt werden.

Bei einer ***lexikalischen Analyse*** wird ***plain text*** in Folgen von logisch zusammengehörigen Einheiten (beispielsweise in Folgen von Zeichen), sogenannte ***Tokens***, zerlegt. Bei der ***grammatischen Analyse*** werden Eingabedaten auf ***syntaktische*** Regeln überprüft, an die sie sich halten.[107] Folgendes Beispiel veranschaulicht dies:

Ein ***Parser*** würde, um bei der obig bereits angesprochenen bei den Sprechstunden auftretenden Problematik zu bleiben, die Struktur

> Vorname Nachname
> Sprechstunde

auch in

> Vorname Nachname
> Sprechstunde
> Der erste Donnerstag jedes Monats entfällt aufgrund von Fortbildung!

erkennen und feststellen, dass die dritte Zeile dieser Struktur nicht entspricht. Die Struktur der zu extrahierenden Daten könnte dem ***Parser***, selbstverständlich nicht wie hier in deutscher Sprache, sondern in einer Programmiersprache, wie folgt beschrieben werden:

- Zeilen, die einen Vornamen und einen Nachnamen beinhalten, bestehen aus genau zwei Wörtern. Diese sind durch ein Leerzeichen getrennt.
- Sprechstunden sind im Format Wochentag, Stunde (Uhrzeit) beschrieben.

Die dritte Zeile der unregelmäßigen Struktur entspricht dieser Beschreibung nicht und würde daher nicht erkannt werden.

[107] vgl. http://www.iue.tuwien.ac.at/phd/demel/node107.html (19.01.2016)

Bei der Datenextraktion aus ***Webseiten*** ist zumeist eine ***HTML***-Interpretation notwendig. Es wird folglich ein ***HTML-Parser*** benötigt. Ein Beispiel hierfür ist der in der Programmiersprache ***Python***[108] geschriebenen `html.parser`.[109]

Reicht selbst dieser nicht aus, um brauchbare Daten zu erhalten, besteht die Möglichkeit, selbst einen auf die jeweilige Anwendung zugeschnittenen ***Parser*** zu schreiben.

Diese Unternehmung wird durch Programme wie ***Bison***[110] und ***Flex***[111] unterstützt. Diese werden im Rahmen dieser Arbeit jedoch nicht besprochen.

[108] vgl. https://www.python.org/ (19.01.2016)
[109] vgl. https://docs.python.org/2/library/htmlparser.html (19.01.2016)
[110] vgl. https://www.gnu.org/software/bison/ (19.01.2016)
[111] vgl. http://flex.sourceforge.net/ (19.01.2016)

6 Fallbeispiele

Nach der theoretischen Analyse der einzelnen *Tools* folgt in diesem Teil die praktische Anwendung jener. Die hier angeführten Fallbeispiele wurden speziell für die besprochenen *Tools* erarbeitet und werden im Folgenden Schritt für Schritt erklärt.

6.1 Extraktion der Sprechstundenliste des BRG19

Das erste Fallbeispiel ist die Extraktion der Sprechstundenliste des BRG19 aus dessen *Homepage*. Ziel ist es, diese als Datei im Format *Comma-separated values* (*CSV*) zu erhalten. Das *CSV*-Format wird dafür verwendet, um Dateien zwischen diversen Tabellenkalkulationsprogrammen (wie zum Beispiel *Microsoft Excel*[112]) auszutauschen und zu konvertieren. Es ist im *RFC* 4180[113] grundlegend beschrieben.

Die Sprechstundenliste war am 20. Jänner 2016 unter

```
http://www.brg19.at/public.php/27/2
```

abzurufen. Der erste Schritt ist, diese *Webseite* im *HTML*-Format herunterzuladen, um sie bearbeiten zu können. Diese Aufgabe übernimmt in diesem Fall `wget`, sie könnte allerdings auch in äquivalenter Art und Weise mit `curl` umgesetzt werden.

Folgende Eingabe in ein *Terminal* erfolgt:

```
wget --output-document=sprechstundenliste.html http://www.brg19.at/public.php/27/2
```

Die Datei liegt nun im *Arbeitsverzeichnis*.

Durch das Mitliefern des entsprechenden *Parameters* wurde der heruntergeladenen Datei ein aussagekräftiger Dateiname, in diesem Fall der Name `sprechstundenliste.html` zugewiesen.

Aufgrund der Länge der Datei ist diese auf dem beiliegenden Speichermedium vorhanden. Der *Quelltext* kann im *Browser Mozilla Firefox* durch folgende Eingabe in die *Adresszeile* eingesehen werden:

```
view-source:http://www.brg19.at/public.php/27/2
```

Die Sprechstundenliste liegt in Tabellenform im *HTML*-Format vor.

[112] vgl. https://products.office.com/de-at/excel (19.01.2016)
[113] vgl. https://tools.ietf.org/html/rfc4180#page-2 (19.01.2016)

Dabei sieht die Tabellenzeile einer Lehrperson im *Browser* dargestellt wie folgt aus:

(Die Spalten beinhalten chronologisch: Name, Email: @brg19.at, Wochentag, Zeit, Ort)

SCHÖBEL Dipl.-Ing. Christian	scoe	Dienstag	7. Stunde (13:50 - 14:40)	Informatik Sammlung

Abbildung 15: Tabellenzeile einer Lehrperson

Im *HTML*-Format sieht diese Zeile so aus:

```
1  <tr bgcolor="#e0e0ff">
2    <td>SCHÖBEL Dipl.-Ing. Christian</td>
3    <td>scoe</td>
4    <td>Dienstag</td>
5    <td>7. Stunde <br />(13:50 - 14:40)</td>
6    <td>Informatik Sammlung</td>
7  </tr>
```

Quelltext 9: Aufbau der Tabellenzeile einer Lehrperson

Der *HTML-Tag* `<tr>` beschreibt eine Zeile. `<td>` beschreibt die Spalten, die sich in dieser Zeile befinden.

Ziel ist es, die extrahierten Daten zur Weiterverarbeitung in ein für die Anwendungszwecke geeignetes Format umzuwandeln, in diesem Fall in das bereits erwähnte *CSV*-Format.

Die extrahierten Daten werden über das *Terminal* mithilfe der *GNU Coreutils* und `sed` weiterverarbeitet. Die endgültige *Befehlskette*, die wegen der Übersichtlichkeit in Zeilen unterteilt wurde und im Folgenden zeilenweise erklärt wird, sieht wie folgt aus:

```
1  cat sprechstundenliste.html |
2    tr '\n' ' ' |
3    sed 's#</tr>#\n#g' |
4    cut -c 32- |
5    sed 's#</td>##g' |
6    sed 's#<td>#;#g' |
7    sed 's#\t##g' |
8    tail -n +7 |
9    head -n -2 |
10   sed 's#<br />##g' |
11   sed 's#<td colspan="2">#;;#g' |
12   sed 's# ;#;#g'
13   > sprechstundenliste.csv
```

Quelltext 10: Befehlskette zur Extraktion der Sprechstundenliste

Gearbeitet wird hierbei mit sogenannten ***Pipelines*** (***Pipes***). Diese ermöglichen der Anwenderin beziehungsweise dem Anwender die Ausgabe eines Programms in die Eingabe eines anderen Programms umzuleiten. Die ***Pipelines*** werden durch das Zeichen „|“ ausgezeichnet. Sie sind im zuvor abgebildeten ***Quelltext*** am Ende der jeweiligen Zeile angeführt.

Zeile 1: `cat sprechstundenliste.html |`

`cat` würde hierbei die Datei auf dem Bildschirm als Text ausgeben. Diese Ausgabe wird durch eine ***Pipeline*** (`|`) zum nächsten ***Befehl*** umgeleitet. Diese Umleitung erfolgt bei den weiteren ***Befehlen*** ebenfalls, wird im Folgenden aber nicht mehr gesondert erwähnt.

Zeile 2: `tr '\n' ' '`

`tr` ersetzt alle `'\n'`, also alle Zeilenumbrüche, durch ein Leerzeichen, da zwischen den zwei Hochkommata ein Leerzeichen ( ) steht.

Zeile 3: `sed 's#</tr>#\n#g'`

`sed` ersetzt alle schließenden `<tr>`-***Tags*** (`</tr>`) durch einen Zeilenumbruch (`\n`). Dies hat zur Folge, dass nun jede Lehrperson eine eigene Zeile hat.

Zeile 4: `cut -c 32-`

`cut` schneidet in jeder Zeile den Text ab inklusive dem 32. Zeichen aus. Diese sind speziell für diese Anwendung abgezählt und somit beginnt eine neue Zeile genau mit dem Namen der Lehrperson.

Zeile 5: `sed 's#</td>##g'`

`sed` ersetzt nun alle schließenden `<td>`-***Tags*** (`</td>`) durch einen Leerstring, da zwischen den zwei ***Delimitern*** kein Zeichen angeführt ist. Dadurch werden diese ***Tags*** entfernt.

Zeile 6: `sed 's#<td>#;#g'`

Nun werden alle öffnenden `<td>`-***Tags*** (`<td>`) von `sed` durch ein Semikolon (`;`), in diesem Fall das Trennzeichen der ***CSV***-Datei, ersetzt.

Zeile 7: `sed 's#\t##g'`

Von `sed` werden alle Tabstopps (`\t`) entfernt.

Zeile 8: `tail -n +7`

`tail` gibt den gesamten Text ab inklusive der siebten Zeile aus.

Zeile 9: `head -n -2`

`head` gibt den gesamten Text exklusive der letzten zwei Zeilen aus.

Durch die Verwendung von `tail` und `head` werden durch die Extraktion überflüssig gewordene Zeilen, wie zum Beispiel jene, die den *HTML*-Kopf beinhalten, entfernt. Ab beziehungsweise bis zu welcher Zeile ausgegeben werden soll, ist abgezählt.

Zeile 10: `sed 's#<br />##g'`
`sed` entfernt nun noch die durch die Extraktion überflüssig gewordenen `<br />`-*Tags*.

Zeile 11: `sed 's#<td colspan="2">#;;#g'`
`sed` ersetzt den nun überflüssigen *Tag* `<td colspan="2">` durch zwei Semikolons. Dies ist für den Erhalt der Tabellenstruktur erforderlich, da die Sprechstunde der Frau Direktorin einen Ausnahmefall darstellt, da für diese kein fixer Termin festgelegt ist.

Zeile 12: `sed 's# ;#;#g'`
`sed` sorgt dafür, dass jedes Leerzeichen, das vor einem Semikolon steht, entfernt wird.

Zeile 13: `> sprechstundenliste.csv`
Das Zeichen „>“ leitet die *Standardausgabe* statt auf den Bildschirm in die Datei `sprechstundenliste.csv` um, die im *Arbeitsverzeichnis* angelegt wird. Die Zeile einer Lehrperson sieht nun wie folgt aus:

```
SCHÖBEL Dipl.-Ing. Christian;scoe;Dienstag;7. Stunde (13:50 - 14:40);Informatik Sammlung
```

Quelltext 11: Tabellenzeile einer Lehrperson (`sprechstundenliste.csv`)

Diese Struktur entspricht exakt der bereits in Abbildung 15 dargestellten, jedoch ist sie nun nicht im *HTML*-, sondern im *CSV*-Format abgebildet. Die angewendete *Befehlskette* kann über ein *Skript* vollständig automatisiert werden.

Eine etwas andere Vorgangsweise ist beim Download mit dem textbasierten *Browser* `lynx` und der darauffolgenden Weiterverarbeitung bei der Extraktion einer Liste der Lehrpersonen aus der gleichen Quelldatei zu verzeichnen. Der *Quelltext* hierzu ist im Anhang (A.II) angeführt.

6.2 Extraktion der eigenen Modulliste von elischa.brg19.at

Das zweite Fallbeispiel befasst sich mit der Extraktion der eigenen Modulliste aus dem Portal „Elischa“ (elischa.brg19.at) des BRG19. Diese erweist sich als etwas komplizierter, da die eigene Modulliste erst nach dem ***Einloggen*** in das Portal verfügbar ist. Für das automatisierte ***Einloggen*** und den Download der Modulliste wird das ***Tool Selenium*** zur Anwendung gebracht.

Folgende ***Befehls***abfolge ist hierfür erforderlich:

Befehl	Ziel	Wert
`open`	`/login.php`	
`type`	`name=username`	`vrecrafa`
`type`	`name=password`	`1234strengGEHEIM#`
`clickAndWait`	`css=input[type="submit"]`	
`open`	`/pupils/elias/elias-angemeldet.php`	
…	…	…

Tabelle 10: Befehle zur Extraktion der Modulliste im HTML-Format

Zunächst wird unter elischa.brg19.at die Datei `login.php` geöffnet. Dieser werden der Account-Name, in diesem Fall „vrecrafa“, und das Passwort übergeben. Danach wird der Button mit der Aufschrift „anmelden!“ geklickt, um ***eingeloggt*** zu werden.

Abbildung 16: Diese Eingaben werden von Selenium automatisiert

Nach dem erfolgreichen *Einloggen* wird die unter

`http://elischa.brg19.at/pupils/elias/elias-angemeldet.php`

abzurufende Datei geöffnet. Diese beinhaltet die Modulliste des dazugehörigen Accounts, in diesem Fall jene von „vrecrafa".

Für die Speicherung der Datei ist *Selenium IDE* in diesem Fall nicht geeignet. Um die Datei dennoch zu speichern, wird die bereits erwähnte Möglichkeit des Exportierens in der *Syntax* einer Programmiersprache, in diesem Fall *Python*[114], verwendet. Das vollständige *Python-Skript* ist im Anhang (A.III) angeführt. Jene Passagen, die nicht von *Selenium IDE* generiert wurden, sind **`fett`** gedruckt. Im Folgenden werden sie kurz besprochen.

```
27  # Pfadangabe - Wo soll Datei gespeichert werden?
28  f = codecs.open("C:/frei/" + "modulliste" + ".html", "w", "utf-8")
29  f.write(driver.page_source)
30  f.close()
```

Quelltext 12: Befehle zur Speicherung des Quelltexts der Modulliste

- Zeile 27 ist ein Kommentar, dieser wird bei der Programmausführung nicht berücksichtigt.[115]
- Zeile 28 öffnet beziehungsweise erstellt die Datei, in welcher der *Quelltext* der Modulliste gespeichert werden soll. Die Datei soll in diesem Fall unter `C:/frei/` gespeichert werden (*Pfad*angabe hier mit *Slashes*), den Namen `modulliste` tragen und die Endung `.html` haben. Weiter soll die Datei mit Schreibrechten geöffnet werden, sowie in *UTF-8*[116] codiert sein.[117]
- Zeile 29 sorgt dafür, dass der *Quelltext* der Modulliste in diese Datei geschrieben wird.[118] Dabei kommt der *Selenium WebDriver*[119], ein *Tool* aus der Ansammlung der *Selenium-Tools*, zur Anwendung.
- Zeile 30 veranlasst das Schließen der geöffneten Datei.[120]

[114] vgl. https://www.python.org/ (19.01.2016)
[115] vgl. https://www.python.org/dev/peps/pep-0008/#inline-comments (19.01.2016)
[116] vgl. https://tools.ietf.org/html/rfc3629 (19.01.2016)
[117] vgl. https://docs.python.org/2/library/codecs.html (19.01.2016)
[118] vgl. https://docs.python.org/2/tutorial/inputoutput.html (19.01.2016)
[119] vgl. http://www.seleniumhq.org/projects/webdriver/ (19.01.2016)
[120] vgl. https://docs.python.org/2/tutorial/inputoutput.html (19.01.2016)

Die Datei ist nach erfolgreichem Download im von der Benutzerin beziehungsweise dem Benutzer gewählten *Verzeichnis* abgelegt. Aufgrund der Länge der Datei wird sinnvollerweise lediglich mit Ausschnitten gearbeitet. Die Datei ist auf dem beiliegenden Speichermedium vorhanden.

Auch die Modulliste liegt in Tabellenform im *HTML*-Format vor. Sie sieht im *Browser* dargestellt wie folgt aus:

#	Modul-ID	Fächer	Matura	Titel	LehrerInnen	Zeit	Optionen/Status
21	15225	M		Kompetent in Mathematik	DMA	Mi, 14:40 - 16:20	
20	15144	SQ	SQ	Selbstmanagement	POB	Mi, 14:40 - 16:20	
19	15131	PE	PE	Psychotherapien	POB	Mo, 14:40 - 16:20	
18	15123	INF		Projektarbeit aus Informatik	SCOE	Di, 16:20 - 18:00	
17	15120	INF	INF	Informatik 1: PASCAL für Anfänger	SCOE	Di, 14:40 - 16:20	
16	14229	M		Mathematik Kompetent	DMA,FRA,DOV	Di, 16:20 - 18:00	
15	14222	INF	INF	PASCAL für Fortgeschrittene	SCOE	Di, 14:40 - 16:20	
14	14220	H	H	Lebensformen im Mittelalter	JLK	Mo, 16:20 - 18:00	
13	14208	BIO	BIO	Meeresbiologie	SCL	Mo, 14:40 - 16:20	
12	14129	L	L	Rom als Keimzelle Europas	DOB	Mo, 14:40 - 16:20	
11	14127	INF	INF	Computernetzwerke	SCOE	Di, 16:20 - 18:00	
10	14126	INF	INF	Einführung in Linux	SCOE	Di, 14:40 - 16:20	
9	14125	INF	INF	Projektarbeit aus Informatik	SCOE	Do, 16:20 - 18:00	
8	13251	SP	SP	Spanisch 2		nach Vereinbarung	
7	13247	SQ	SQ	Konflikte verstehen und bearbeiten	LED	Mi, 14:40 - 16:20	
6	13220	INF	INF	Webscripting und Datenbanken	SCOE	Do, 16:20 - 18:00	
5	13219	INF	INF	Betriebssysteme	SCOE	Do, 14:40 - 16:20	
4	13158	SP	SP	Spanisch 1		nach Vereinbarung	
3	13128	INF	INF	Maschinennahe Programmierung	SCOE	Do, 14:40 - 16:20	
2	13127	INF	INF	Microcontroller	SCOE	Do, 16:20 - 18:00	
1	13108	CH	CH	Chemische Übungen	STN	Mo, 14:40 - 16:20	

Abbildung 17: Modulliste

Die Tabellenzeile eines Moduls sieht im *HTML*-Format so aus:

```
1  <tr class="hi1">
2    <td class="num">10</td>
3    <td class="mid">14126</td>
4    <td class="fks">INF</td>
5    <td class="fks">INF</td>
6    <td>Einführung in Linux</td>
7    <td>SCOE</td>
8    <td class="tright"><span class="nobr">Di, 14:40 - 16:20</span></td>
9    <td><span class="nobr"></span></td>
10 </tr>
```

Quelltext 13: Aufbau der Tabellenzeile eines Moduls

Das Ziel ist auch hierbei, die extrahierten Daten in ein an die Anwendungszwecke angepasstes Format zur Weiterverarbeitung umzuwandeln. Auch in diesem Fall ist das, wie schon im ersten Fallbeispiel, das bereits erwähnte *CSV*-Format.

Auch in diesem Fallbeispiel werden die extrahierten Daten mithilfe der *GNU Coreutils* weiterverarbeitet.

Erneut die endgültige *Befehlskette* in durch Zeilen separierter Form:

```
 1  cat modulliste.html |
 2    tr '\n' ' ' |
 3    sed 's#</tr>#\n#g' |
 4    sed 's#<td[^>]*>##' |
 5    sed 's#<td[^>]*>#;#g' |
 6    sed 's#<[^>]*>##g' |
 7    sed 's#\t##g' |
 8    cut -c 3- |
 9    sed 's# ;#;#g' |
10    tail -n +2
11    > modulliste.csv
```

Quelltext 14: Befehlskette zur Extraktion der Modulliste

Im Folgenden werden nun allerdings nur mehr jene Zeilen besprochen, die im ersten Fallbeispiel noch nicht in dieser Form erwähnt wurden.

Folgende Zeilen werden nicht mehr erklärt:

- Zeile 1, da sie bis auf den Dateinamen mit Zeile 1 aus Fallbeispiel 1 ident ist
- Zeilen 2, 3 und 7, da sie ident mit selbigen aus Fallbeispiel 1 sind
- Zeile 9, da sie ident mit Zeile 12 aus Fallbeispiel 1 ist

Folgende Zeilen werden gesondert erklärt:

Zeile 4: `sed 's#<td[^>]*>##'`

`sed` entfernt den ersten `<td>`-*Tag*, mitsamt dessen *Parametern*.

Da *HTML-Tags* verschieden aussehen, sich aber alle an eine bestimmte Logik halten, wird hierbei mit *Regular Expressions*[121] (*Reguläre Ausdrücke*) gearbeitet.

[121] vgl. u. a. http://pubs.opengroup.org/onlinepubs/009695399/basedefs/xbd_chap09.html (19.01.2016)

Folgendes Beispiel veranschaulicht die Vorgehensweise bei der Arbeit mit ***Regular Expressions***:

```
<h1>

<p>
```

Die beiden angeführten *HTML-Tags* haben folgende Gemeinsamkeiten:

- Sie sind beide öffnende *Tags*, das heißt, sie haben beide kein „/“.
- Sie beginnen mit <.
- Sie enden mit >.

Eine *Regular Expression* beschreibt diese Gemeinsamkeiten. Dies passiert auch im obigen *Befehl*.

Folgende Beschreibung trifft der Ausdruck `<td[^>]*>` bei der Verwendung mit `sed`:[122]

- Der *HTML-Tag* wird mit < geöffnet, danach folgt `td`.
- `[^>]` wählt alle Zeichen aus, die kein > sind. Das ist wichtig, da weitere *Parameter* vorhanden sein könnten.
- * bedeutet, dass die ausgewählten Zeichen beliebig oft vorkommen dürfen.
- Der *HTML-Tag* wird mit > geschlossen.

In jeder Zeile wird die erste ***Zeichenkette***, die dieser Beschreibung entspricht, entfernt, alle weiteren jedoch nicht, da die *Option* für `g` (für „global“) nicht aktiviert wurde.

Zeile 5: `sed 's#<td[^>]*>#;#g'`
`sed` ersetzt nun global alle weiteren `<td>`-*Tags* durch Semikolons, in diesem Fall das Trennzeichen der *CSV*-Datei.

Zeile 6: `sed 's#<[^>]*>##g'`
`sed` entfernt global alle weiteren *HTML-Tags*. Der einzige Unterschied verglichen mit der Beschreibung in Zeile 4 liegt darin, dass nach dem Zeichen „<“ nicht auch noch die ***Zeichenfolge*** „`td`“ stehen muss, sondern beliebige Zeichen stehen dürfen.

[122] vgl. https://www.gnu.org/software/sed/manual/html_node/Regular-Expressions.html (20.01.2016)

Zeile 8: `cut -c 3-`

`cut` schneidet den Text ab inklusive dem dritten Zeichen aus. Diese sind speziell für diesen Fall abgezählt.

Zeile 10: `tail -n +2`

`tail` gibt den gesamten Text ab inklusive der zweiten Zeile aus.

Zeile 11: `> modulliste.csv`

Das Ergebnis der ***Befehlskette*** wird in der Datei `modulliste.csv` gespeichert:

```
10;14126;INF;INF;Einführung in Linux;SCOE;Di, 14:40 - 16:20;
```

Quelltext 15: Tabellenzeile eines Moduls (`modulliste.csv`)

Auch hier liegt nun die gleiche Struktur vor, die in Abbildung 17 bereits dargestellt worden ist. Die verwendete ***Befehlskette*** kann auch hier über ein ***Skript*** vollständig automatisiert werden.

7 Bewertungen und Schlussfolgerungen

In dieser Arbeit wurde der gesamte Prozess einer automatisierten Datenextraktion aus *Webseiten* abgehandelt. Daraus lassen sich einige Schlussfolgerungen ziehen, die im folgenden Kapitel abgehandelt werden.

7.1 Kenntnissen über das Internetsurfen und den Aufbau von Webseiten

Grundsätzlich ist es wichtig, über die Vorgänge, die beim *Internetsurfen* ablaufen, Bescheid zu wissen, um den Prozess der automatisierten Datenextraktion nachvollziehen zu können. Ist kein Wissen über die theoretische Funktionsweise des *HTTP* oder den grundlegenden Aufbau einer *Webseite* vorhanden, ist es de facto unmöglich, einen Automatisierungsprozess zu planen, da der Aufbau einer *Webseite* und der Transport dieser unabdingbar für die Datenextraktion aus dieser sind. Die wohl wichtigste Grundlage ist in diesem Zusammenhang die Tatsache, dass die Daten, welche man aus einer *Webseite* extrahieren will, in der Regel im *HTML*-Format vorliegen. Insofern sind detaillierte Kenntnisse über *HTML* eine essenzielle Voraussetzung für eine automatisierte Datenextraktion aus *Webseiten*. Die einzig noch sinnvoll erscheinende Möglichkeit, *HTML* zu umgehen, ist jene, die *Webseite* nicht mithilfe von `wget` oder `curl` herunterzuladen, sondern einen textbasierten *Browser* wie beispielsweise `lynx` zu verwenden, da dieser die Seite als Text ohne die *HTML*-Struktur abliefern kann.

7.2 Die Art einer Webseite

Neben dem Aufbau einer *Webseite* ist auch deren damit in Verbindung stehende Art für die automatisierte Datenextraktion von großer Bedeutung. Diese Arbeit nahm empirisch eine eigene Klassifizierung von *Webseiten* vor. In diesem Zusammenhang ist zu sagen, dass die eine automatisierte Datenextraktion planende Person über die grundlegende Funktionsweise und die Art der jeweiligen *Webseite* Bescheid wissen muss. Mutmaßungen über die Art einer *Webseite* lassen sich in jedem Fall leicht anstellen. Beispielsweise ist anzunehmen, dass *Webseiten*, die Informationen wie das heutige Datum oder Ähnliches beinhalten, generiert werden. Ob diese Elemente *server*seitig oder *client*seitig generiert werden, lässt sich leicht feststellen, indem man sich den erhaltenen *Quelltext* ansieht. Sind darin keine *client*seitig generierten Elemente enthalten, ist festzustellen, dass die *Webseite* statisch ist oder *server*seitig generiert worden ist. Sofern die *Webseite* keinen *Login-Bereich hat*, ist eine weitere Unterscheidung zwischen den beiden genannten Arten für die Anwenderin beziehungsweise den Anwender sogar irrelevant, da sich die besprochenen *Tools* für

beide Arten gleich verhalten. Die Extraktion von Daten aus *Webseiten*, die durch einen *Login-Bereich* geschützt sind, lässt sich hingegen sinnvollerweise nur mit *Selenium* realisieren.

7.3 Sinnvolle Verwendung geeigneter Tools

In diesem Zusammenhang ist generell zu sagen, dass die wohl wichtigste Entscheidung im Prozess der automatisierten Datenextraktion jene über die verwendeten *Tools* ist. Eine unbedachte Wahl kann hierbei auf der einen Seite einen enormen Mehraufwand und auf der anderen Seite eine geringere Zuverlässigkeit, was die Korrektheit der extrahierten Daten betrifft, bedeuten. Insofern ist es notwendig, grundlegende Überlegungen über die einzelnen *Tools* anzustellen und zu ermitteln, ob sie wirklich die Anforderungen für den entsprechenden Anwendungsbereich erfüllen. Weiter ist hierbei zu bemerken, dass die *GNU Coreutils* grundsätzlich mit wenig Aufwand sehr schnell brauchbare Ergebnisse liefern. Bei einer komplexeren Struktur der Quelle stoßen sie allerdings schnell an ihre Grenzen. Abhilfe kann man sich vorerst noch schaffen, indem man mit *Regular Expressions* arbeitet. Diese gehen aber ohnehin schon in Richtung *Parser*, da sie ebenfalls gewisse Vorkommnisse allgemein beschreiben. Oftmals gibt es aber wohl kaum einen anderen Ausweg, als einen *Parser* zu verwenden beziehungsweise sogar eigenhändig zu schreiben.

7.4 Abschlussbewertung

Abschließend lässt sich feststellen, dass die automatisierte Datenextraktion aus *Webseiten* in vielen Fällen erstrebenswert scheint, da sie eine große Zeitersparnis bedeutet und vor allem auch die Fehleranfälligkeit reduziert. Es ist jedoch zu bemerken, dass ein großes Maß an Vorkenntnissen benötigt wird, um eine zuverlässige Anwendung zu gewährleisten. Es kommt dabei darauf an, inwiefern der dafür erforderliche Aufwand in einem Verhältnis zum daraus resultierenden Nutzen steht. Eine Automatisierung macht nur dann Sinn, wenn dadurch merklich Zeit eingespart werden kann und die extrahierten Daten regelmäßig in einer aktuellen Form benötigt werden. Ansonsten steht der Aufwand wohl kaum im Verhältnis zum tatsächlichen Nutzen. Die Automatisierung einer einmaligen Extraktion von Daten aus einer bestimmten *Webseite* scheint ohne fundierte Vorkenntnisse kaum sinnvoll. Für eine wiederkehrende Extraktion ist es allerdings sehr wohl empfehlenswert, sich entsprechende Kenntnisse anzueignen und eine Automatisierung anzustreben.

A Anhang

A.I Abstract (deutschsprachige Fassung)

Die automatisierte Datenextraktion aus *Webseiten* hat das Ziel der Zeitersparnis und der Verringerung der Fehlerquote. Diese Arbeit befasst sich mit einem derartigen Automatisierungsprozess.

Begonnen wird mit der Beschreibung der Vorgänge beim *Internetsurfen*. Dabei wird auf das *Hypertext Transfer Protocol* (*HTTP*) und speziell auf dessen Funktionseinheiten *Client* und *Server* eingegangen. Weiter wird empirisch eine Klassifizierung von *Webseiten* vorgenommen. In weiterer Folge werden diverse *Tools*, welche für die Automatisierung erforderlich sind, besprochen. Diese werden in die Kategorien „Tools zum Abruf" und „Tools zur Verarbeitung" eingeteilt.

Nach der theoretischen Beschreibung der einzelnen *Tools* folgt die Analyse eines Automatisierungsprozesses auf Basis zweier Fallbeispiele. Dabei werden die Sprechstundenliste des BRG19 sowie eine individuelle Liste an gewählten Modulen aus den im Vorfeld heruntergeladenen Dateien im Format *Hypertext Markup Language* (*HTML*) extrahiert und in Dateien im Format *Comma-separated values* (*CSV*) umgewandelt.

Abschließend werden Bewertungen und Schlussfolgerungen den Automatisierungsprozess betreffend angestellt, die analysieren, welche Kenntnisse für eine derartige Automatisierung erforderlich sind und inwiefern eine Automatisierung sinnvoll beziehungsweise nicht sinnvoll ist.

A.II Fallbeispiel 1, Daten im Textformat von **lynx** abgeholt

```
lynx --dump http://www.brg19.at/public.php/27/2 > lynxdump.txt
```

```
1  cat lynxdump.txt |
2    cut -c 4- |
3    head -n -17 |
4    tail -n +15 |
5    sort |
6    sed 's/\s[a-z]/\n/' |
7    grep -v '^[a-z]' |
8    tail -n +71
9    > lehrpersonenliste_lynx.txt
```

Quelltext 16: Befehlskette zur Extraktion einer Liste der Lehrpersonen (lynx)

Die Datei `lynxdump.txt` ist auf dem beiliegenden Speichermedium vorhanden.

A.III Python-Skript zur Speicherung der Modulliste

```
1  # -*- coding: utf-8 -*-
2  from selenium import webdriver
3  from selenium.webdriver.common.by import By
4  from selenium.webdriver.common.keys import Keys
5  from selenium.webdriver.support.ui import Select
6  from selenium.common.exceptions import NoSuchElementException
7  from selenium.common.exceptions import NoAlertPresentException
8  import unittest, time, re, codecs
9
10 class Webdriver(unittest.TestCase):
11     def setUp(self):
12         self.driver = webdriver.Firefox()
13         self.driver.implicitly_wait(30)
14         self.base_url = "http://elischa.brg19.at/"
15         self.verificationErrors = []
16         self.accept_next_alert = True
17
18     def test_webdriver(self):
19         driver = self.driver
20         driver.get(self.base_url + "/login.php")
21         driver.find_element_by_name("username").clear()
22         driver.find_element_by_name("username").send_keys("vrecrafa")
23         driver.find_element_by_name("password").clear()
24         driver.find_element_by_name("password").send_keys("[redacted]")
25         driver.find_element_by_css_selector("input[type=\"submit\"]").click()
26         driver.get(self.base_url + "/pupils/elias/elias-angemeldet.php")
27         # Pfadangabe - Wo soll Datei gespeichert werden?
28         f = codecs.open("C:/frei/" + "modulliste" + ".html", "w", "utf-8")
29         f.write(driver.page_source)
30         f.close()
31
32     def is_element_present(self, how, what):
```

```
33          try: self.driver.find_element(by=how, value=what)
34          except NoSuchElementException as e: return False
35          return True
36      def is_alert_present(self):
37          try: self.driver.switch_to_alert()
38          except NoAlertPresentException as e: return False
39          return True
40      def close_alert_and_get_its_text(self):
41          try:
42              alert = self.driver.switch_to_alert()
43              alert_text = alert.text
44              if self.accept_next_alert:
45                  alert.accept()
46              else:
47                  alert.dismiss()
48              return alert_text
49          finally: self.accept_next_alert = True
50
51      def tearDown(self):
52          finally: self.accept_next_alert = True
53
54          self.driver.quit()
55          self.assertEqual([], self.verificationErrors)
56
57  if __name__ == "__main__":
58      unittest.main()
```

Quelltext 17: Python-Skript zur Speicherung der Modulliste

B Glossar

Im folgenden Glossar sind alle im Text gekennzeichneten Begriffe alphabetisch aufsteigend sortiert. Auf der linken Seite ist jeweils der Begriff, auf der rechten Seite die Erklärung zu finden. Die Quellen wurden als Fußnoten angefügt und sind auch im Literaturverzeichnis (C Literaturverzeichnis) angeführt. Die Voranstellung von „Abk." kennzeichnet Abkürzungen als solche. Mehrfachbedeutungen wurden durch Voranstellung von „hier:" ausgewiesen, worauf die im Kontext dieser Arbeit relevante Bedeutung folgt. Verweise wurden durch das die Voranstellung von „siehe" gekennzeichnet. Referenzen auf das Glossar wurden nach dem bisher gebrauchten Schema auf der rechten, nicht aber auf der linken Seite gesetzt.

In einem Glossar ist es teilweise notwendig, Begriffe wörtlich oder zumindest beinahe wörtlich zu übernehmen. Da aber jeweils nur die im Kontext relevante Bedeutung des Begriffs, teilweise sogar in leicht abgewandelter Form, übernommen wurde, wurde auch hierbei mit der in der Arbeit gebräuchlichen Zitierweise gearbeitet.

Abk. ASCII	siehe *American Standard Code for Information Interchange*
Abk. ASP.NET	siehe *Active Server Pages .NET*
Abk. CSS	siehe *Cascading Style Sheets*
Abk. CSV	siehe *Comma-separated values*
Abk. FTP	siehe *File Transfer Protocol*
Abk. GPL	siehe *GNU General Public License*
Abk. HTML	siehe *Hypertext Markup Language*
Abk. HTTP	siehe *Hypertext Transfer Protocol*
Abk. IETF	siehe *Internet Engineering Task Force*
Abk. IMAP	siehe *Internet Message Access Protocol*
Abk. JSP	siehe *JavaServer Pages*
Abk. LDAP	siehe *Lightweight Directory Access Protocol*
Abk. NTP	siehe *Network Time Protocol*
Abk. PHP	rekursives Akronym für *PHP Hypertext Processor*; siehe *PHP Hypertext Processor*
Abk. POP	siehe *Post Office Protocol*
Abk. RFC	siehe *Request for Comments*
Abk. SMTP	siehe *Simple Mail Transfer Protocol*
Abk. URL	siehe *Uniform Resource Locator*
Abk. W3C	siehe *World Wide Web Consortium*
Active Server Pages .NET	Software zur Entwicklung *dynamischer Webseiten*[123]
Addon	Hilfsprogramm, mit welchem ein Anwendungsprogramm erweitert wird[124]

[123] vgl. http://www.asp.net/ (19.01.2016)
[124] vgl. http://www.duden.de/rechtschreibung/Add_on (19.01.2016)

Adresse	hier: siehe *Internetadresse*
Adresszeile	Bedienelement eines *Browsers*, in welches unter anderem die *URL* einer *Webseite* eingegeben werden kann[125]
American Standard Code for Information Interchange	genormter Code für Ziffern, Buchstaben und Sonderzeichen[126]
Anfrage	hier: siehe *Request*
Anfragenachricht	siehe *Request-Message*
Antwort	hier: siehe *Request*
Arbeitsverzeichnis	siehe *Verzeichnis*; *Verzeichnis*, auf dem gearbeitet wird beziehungsweise auf dessen Dateien Operationen angewendet werden sollen[127]
ASCII-Text	Text, der *ASCII* codiert ist[128]
automatisch aktualisierte statische Webseite	*Webseite*, die automatisch aktualisiert wird; Wahl des Aktualisierungszeitpunkts hängt vom Inhalt der *Webseite* ab (Teil der im Rahmen dieser Arbeit getroffenen Klassifizierung von *Webseiten*)
Bash	Shell (= Benutzerschnittstelle, die eingegebene Befehle interpretiert) des GNU-Projekts[129]
Befehl	hier: Anweisung zur Ausführung einer bestimmten Operation an den Computer[130]
Befehlskette	Zusammenkettung von *Befehlen*; siehe *Befehl*
Bison	hier: *Parser*-Generator des GNU-Projekts[131]
Browser	Programm, mit welchem *Webseiten* abgerufen werden können; kann grafische Benutzeroberfläche haben oder textbasiert sein[132]
C	Programmiersprache[133]
Cascading Style Sheets	Sprache zum Festlegen der Darstellung von *HTML*-Dokumenten[134]
`cat`	hier: Programm aus der Ansammlung der *GNU Coreutils* zur Ausgabe des Inhalts einer Textdatei[135]
Client	hier: initiiert Verbindungen zu *Servern*; ist in der Regel ein *Browser*[136]
clientseitig generierte dynamische Webseite	*Webseite*, die beim Abruf vom *Client* generiert wird; Generierung wird vom *Server* gesteuert, erfolgt aber beim *Client*; clientseitig ausgeführter *Quelltext* ist für den *Client* sichtbar (Teil der im Rahmen dieser Arbeit getroffenen Klassifizierung von *Webseiten*)
Comma-separated values	Format, das verwendet wird, um Dateien zwischen diversen Tabellenkalkulationsprogrammen auszutauschen und zu konvertieren[137]
Content-Type	siehe *MIME-Type*

[125] vgl. u. a. https://support.mozilla.org/de/kb/autovervollstaendigung-adressleiste (19.01.2016)
[126] vgl. http://tools.ietf.org/html/rfc20 (22.01.2016)
[127] vgl. u. a. https://www.dict.cc/?s=Arbeitsverzeichnis (22.01.2016)
[128] vgl. http://tools.ietf.org/html/rfc20 (22.01.2016)
[129] vgl. https://www.gnu.org/software/bash/ (19.01.2016)
[130] vgl. http://www.duden.de/rechtschreibung/Befehl#Bedeutung1b (22.01.2016)
[131] vgl. https://www.gnu.org/software/bison/ (19.01.2016)
[132] vgl. http://www.duden.de/rechtschreibung/Browser (22.01.2016)
[133] vgl. u. a. http://csapp.cs.cmu.edu/3e/docs/chistory.html (19.01.2016)
[134] vgl. http://www.w3.org/standards/webdesign/htmlcss#whatcss (19.01.2016)
[135] vgl. https://www.gnu.org/software/coreutils/manual/coreutils.html#cat-invocation (19.01.2016)
[136] vgl. http://tools.ietf.org/html/rfc2616#section-1.3 (19.01.2016)
[137] vgl. https://tools.ietf.org/html/rfc4180#page-2 (19.01.2016)

Cookie	Möglichkeit, *Session*-Information zu codieren; ist ein Datensatz, mit welchem Benutzer einer *Webseite* identifiziert werden kann[138]
Cronjob	automatisiert zeitgesteuert ausgeführte wiederkehrende Aufgabe[139]
`curl`	*Kommandozeilenprogramm* zum Download von *Webseiten*[140]
`cut`	Programm aus der Ansammlung der *GNU Coreutils* zur zeilenweisen Extraktion von Spalten aus einer Eingabe[141]
Dateipfad	Ort, an welchem sich eine Datei befindet[142]
Datenstrom	kontinuierlicher Fluss von Datensätzen[143]
Delimiter	Trennzeichen[144]
Dokumenttyp	Klasse von ähnlichen Dokumenten[145]
dynamische Webseite	*Webseite*, die beim Abruf generiert wird (Teil der im Rahmen dieser Arbeit getroffenen Klassifizierung von *Webseiten*)
Ecma (International)	(European Computer Manufacturers Association); Organisation zur Normung von Informations- und Kommunikationssystemen und Unterhaltungselektronik[146]
ECMAScript	Sprachkern von *JavaScript*[147]
Eingabeparameter	siehe *Parameter*
Einloggen	siehe *Login*
enriched text	*ASCII-Text* mit einfachen Formatierungen[148]
File Transfer Protocol	*Protokoll*, das hauptsächlich für den Transfer von Dateien verwendet wird[149]
Flex	hier: (Beispiel für ein) Programm, das eine *lexikalische Analyse* durchführt[150]
GET	Methode einer *Request-Message*[151]
GIF-Format	(Graphics Interchange Format); Grafikformat für Rasterbilder[152]
GNU Coreutils	Ansammlung von diversen Programmen zur Textverarbeitung[153]
GNU General Public License	Softwarelizenz, die einer Person ermöglicht, Software auszuführen, zu studieren, zu ändern und zu verbreiten (kopieren)[154]
Google Chrome	*Browser* von Google Inc.[155]

[138] vgl. https://tools.ietf.org/html/rfc6265 (19.01.2016) und http://www.duden.de/rechtschreibung/Cookie (22.01.2016)
[139] vgl. u. a. http://pubs.opengroup.org/onlinepubs/9699919799/utilities/crontab.html (19.01.2016)
[140] vgl. http://curl.haxx.se/ (19.01.2016)
[141] vgl. https://www.gnu.org/software/coreutils/manual/coreutils.html#cut-invocation (19.01.2016)
[142] vgl. http://www.linfo.org/path.html (22.01.2016)
[143] vgl. http://www.its.bldrdoc.gov/fs-1037/dir-010/_1451.htm (19.01.2016)
[144] vgl. http://www.its.bldrdoc.gov/fs-1037/dir-011/_1544.htm (22.01.2016)
[145] vgl. http://www.duden.de/rechtschreibung/Dokumenttyp (22.01.2016)
[146] vgl. http://www.ecma-international.org/ (22.01.2016)
[147] vgl. http://ecma-international.org/ecma-262/5.1/ (19.01.2016)
[148] vgl. http://www.iana.org/assignments/media-types/media-types.xhtml (19.01.2016)
[149] vgl. https://tools.ietf.org/html/rfc354 (19.01.2016)
[150] vgl. http://flex.sourceforge.net/ (19.01.2016)
[151] vgl. http://tools.ietf.org/html/rfc2616#section-9.3 (19.01.2016)
[152] vgl. https://www.w3.org/Graphics/GIF/spec-gif89a.txt (23.01.2016)
[153] vgl. http://www.gnu.org/software/coreutils/coreutils.html (22.01.2016)
[154] vgl. u. a. http://www.gnu.org/licenses/gpl-3.0.html
[155] vgl. https://www.google.de/chrome/browser/desktop/index.html (19.01.2016)

Google-Suche	Suche im Internet mithilfe der Suchmaschine Google von Google Inc.[156]
grammatische Analyse	auch *syntaktische Analyse*; Überprüfung von Eingabe auf *syntaktische* Regeln, an die sie sich halten[157]
grep	Programm des GNU-Projekts, das Dateien nach bestimmten Suchmustern durchsucht[158]
head	Programm, das die ersten zehn (oder eine andere Anzahl an) Zeilen einer Textdatei auf dem Bildschirm ausgibt[159]
Hidden Field	deutsch: unsichtbares Formularfeld; Möglichkeit, *Session*-Information unterzubringen[160]
Homepage	beispielsweise die persönliche *Webseite* einer Person[161]
html.parser	in der Programmiersprache *Python* geschriebener *HTML-Parser*[162]
HTML-Parser	*Parser* für Daten im *HTML*-Format; siehe *HTML*; siehe *Parser*
HTML-Tag	siehe *Tag* (Anmerkung: Definition nur auf *HTML* bezogen)
HTTPS	verschlüsselte Variante des *HTTP*; „S“ steht für „secure“ (sicher)[163]
hybride dynamische Webseite	*Webseite*, die sowohl Elemente einer *clientseitig generierten dynamischen Webseite* als auch Elemente einer *serverseitig generierten dynamischen Webseite beinhaltet*; Zugänglichkeit des *Quelltexts* für den *Client* ist elementabhängig; Generierung erfolgt teilweise auf dem *Server*, teilweise beim *Client* (Teil der im Rahmen dieser Arbeit getroffenen Klassifizierung von *Webseiten*)
Hypertext	über *Links* verbundenes Netz aus Bildern und Text, in dem sich Benutzerinnen und Benutzer frei bewegen können[164]
Hypertext Markup Language	Sprache zum Beschreiben der Struktur von aus *Hypertext* bestehenden *Webseiten*[165]
Hypertext Transfer Protocol	*zustandsloses Protokoll* zur Übertragung von Daten über ein *Rechnernetz*[166]
Internet Engineering Task Force	Organisation, welche mit der Weiterentwicklung des Internets befasst ist[167]
Internet Message Access Protocol	*Protokoll*, gemäß welchem E-Mails abgerufen werden[168]
Internetadresse	aus standardisierten Zeichen bestehende Angabe, unter der jemand im Internet erreichbar ist[169]
Internetsurfen	Tätigkeit der wahllosen oder gezielten Suche von Informationen im Internet[170]

[156] vgl. https://www.google.com (19.01.2016)
[157] vgl. http://www.iue.tuwien.ac.at/phd/demel/node107.html (19.01.2016)
[158] vgl. http://www.gnu.org/software/grep/manual/grep.html (19.01.2016)
[159] vgl. https://www.gnu.org/software/coreutils/manual/coreutils.html#head-invocation (19.01.2016)
[160] vgl. http://www.uni-weimar.de/medien/webis/teaching/lecturenotes/web-technology/unit-de-network-protocol3.pdf, Seite 42 (19.01.2016)
[161] vgl. http://www.duden.de/rechtschreibung/Homepage (22.01.2016)
[162] vgl. https://docs.python.org/2/library/htmlparser.html (19.01.2016)
[163] vgl. https://tools.ietf.org/html/rfc2818 (19.01.2016)
[164] vgl. http://www.duden.de/rechtschreibung/Hypertext (22.01.2016)
[165] vgl. http://www.w3.org/standards/webdesign/htmlcss#whathtml (19.01.2016)
[166] vgl. https://tools.ietf.org/html/rfc2616, Abstract (19.01.2016)
[167] vgl. https://www.ietf.org/ (19.01.2016)
[168] vgl. https://tools.ietf.org/html/rfc3501 (19.01.2016)
[169] vgl. http://www.duden.de/rechtschreibung/Internetadresse (22.01.2016)
[170] vgl. http://www.duden.de/rechtschreibung/Internetsurfen (22.01.2016)

JavaScript	Programmiersprache, mit der Webanwendungen realisiert werden können[171]
JavaServer Pages	von Sun Microsystems Inc. entwickelte Programmiersprache zur Realisierung von Webanwendungen[172]
JPEG-Format	Grafikformat für Rasterbilder[173]
Kommandozeile	auf dem Bildschirm gekennzeichnete Stelle, an der man *Befehle* (= Kommandos) zum Starten von Programmen eingeben kann[174]
Kommandozeilenprogramm	Programm ohne grafische Benutzeroberfläche, welches über die *Kommandozeile* bedient wird[175]
lexikalische Analyse	Zerlegung von *plain text* in *Tokens*[176]
Lightweight Directory Access Protocol	*Protokoll* zur Abfrage und Änderung von Informationen in einem *Verzeichnis*dienst[177]
Link	Verknüpfung mit einer anderen Datei oder einer anderen Stelle in der gleichen Datei[178]
Login	deutsch: *Einloggen*[179]
Login-Bereich	Bereich, der ein *Login* erfordert, um Zugriff zu gewähren[180]
lynx	textbasierter *Browser*[181]
manuell aktualisierte statische Webseite	*Webseite*, die manuell verändert wird (Teil der im Rahmen dieser Arbeit getroffenen Klassifizierung von *Webseiten*)
Medientyp	siehe *MIME-Type*
Metadaten	Daten, die Informationen über andere Daten enthalten[182]
Microsoft Edge	*Browser* der Microsoft Corporation[183]
Microsoft Excel	Tabellenkalkulationsprogramm der Microsoft Corporation[184]
Microsoft Windows XP	Betriebssystem der Microsoft Corporation aus dem Jahr 2001[185]
MIME-Type	gibt (bei der Übertragung gemäß *HTTP*) an, um welche Art von Daten es sich handelt[186]
Mozilla Firefox	*Browser* der Mozilla Foundation[187]
Network Time Protocol	*Protokoll*, das für die Synchronisation von Uhren in Computersystemen von Bedeutung ist[188]

[171] vgl. http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf (19.01.2016) und https://developer.mozilla.org/en-US/docs/Web/JavaScript (19.01.2016)
[172] vgl. http://www.oracle.com/technetwork/java/javaee/jsp/index.html (19.01.2016)
[173] vgl. http://jpeg.org/ (23.01.2016)
[174] vgl. http://www.duden.de/rechtschreibung/Kommandozeile (22.01.2016)
[175] vgl. http://www.duden.de/rechtschreibung/Kommandozeile (22.01.2016)
[176] vgl. u. a. https://www.ibm.com/developerworks/community/blogs/nlp/entry/tokenization?lang=en (22.01.2016)
[177] vgl. https://tools.ietf.org/html/rfc4511 (19.01.2016)
[178] vgl. http://www.duden.de/rechtschreibung/Link (22.01.2016)
[179] vgl. http://www.duden.de/rechtschreibung/Log_in (22.01.2016)
[180] vgl. http://www.duden.de/rechtschreibung/Log_in (22.01.2016)
[181] vgl. http://lynx.invisible-island.net/ (19.01.2016)
[182] vgl. https://wiki.selfhtml.org/wiki/HTML/Kopfdaten/meta (22.01.2016)
[183] vgl. https://www.microsoft.com/de-at/windows/microsoft-edge (19.01.2016)
[184] vgl. https://products.office.com/de-at/excel (19.01.2016)
[185] vgl. http://windows.microsoft.com/de-de/windows/windows-help#windows=windows-xp (21.01.2016)
[186] vgl. http://www.w3.org/2001/tag/2002/0129-mime (19.01.2016)
[187] vgl. https://www.mozilla.org/de/firefox/new/ (19.01.2016)
[188] vgl. https://tools.ietf.org/html/rfc5905 (19.01.2016)

Open Source (Open-Source-Software)	Software, deren Quellcode frei zugänglich ist[189]
Opera	*Browser* von Opera Software[190]
Option	siehe *Parameter*
Parameter	für einen Aufruf gültige Einstellungsmöglichkeiten eines Computerprogramms; *HTML*: genauere Spezifikation von *Tags*[191]
Parser	Programm, das eine beliebige Eingabe zerlegt und in ein für die Weiterverarbeitung brauchbares Format umwandelt[192]
Pfad	siehe *Dateipfad*
PHP Hypertext Processor	Programmiersprache, mit der Webanwendungen realisiert werden können[193]
Pipe	siehe *Pipeline*
Pipeline	auch: *Pipe*; ermöglicht Umleitung der Ausgabe eines Programms in die Eingabe eines anderen Programms[194]
plain text	deutsch: einfacher, schlichter Text; Daten, die direkt in Text umgesetzt werden können[195]
POST	Methode einer *Request-Message*[196]
Post Office Protocol	*Protokoll*, gemäß welchem E-Mails abgerufen werden[197]
PostScript-Format	Sprache für Vektorgrafiken, die beschreibt wie Seiten auf Geräten (wie zum Beispiel Druckern) aussehen sollen[198]
Protokoll	hier: Kommunikationsvorschrift; schreibt vor, wie eine Kommunikation abzulaufen hat[199]
Python	hier: Programmiersprache[200]
Quelltext	in einer Programmiersprache geschriebene Abfolge von Programmanweisungen, die vom Menschen gelesen, aber erst nach einer elektronischen Übersetzung vom Computer verarbeitet werden können[201]
Rechnernetz	Vernetzung mehrerer voneinander unabhängiger Rechner, die den Datenaustausch zwischen besagten Rechnern ermöglicht[202]
Regular Expression	Ausdruck zur kontextunabhängigen Beschreibung von *Zeichenketten*[203]
Regulärer Ausdruck	siehe *Regular Expression*

[189] vgl. http://www.duden.de/rechtschreibung/Open_Source_Software (22.01.2016)
[190] vgl. http://www.opera.com/de (19.01.2016)
[191] vgl. u. a. http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2005/n1905.pdf, Seite 16, Kap. 1.3.1 argument (22.01.2016) und http://www.w3.org/standards/webdesign/htmlcss#whathtml (19.01.2016)
[192] vgl. https://www2.informatik.uni-erlangen.de/EN/teaching/SS2012/HalloWelt/PA_2012.pdf, Seite 3 (22.01.2016)
[193] vgl. http://php.net/ (19.01.2016)
[194] vgl. u. a. http://www.linfo.org/pipe.html (22.01.2016)
[195] vgl. https://tools.ietf.org/html/rfc2046#section-4.1.3 (19.01.2016)
[196] vgl. http://tools.ietf.org/html/rfc2616#section-9.5 (19.01.2016)
[197] vgl. https://tools.ietf.org/html/rfc1939 (19.01.2016)
[198] vgl. http://www.adobe.com/devnet/postscript.html (23.01.2016)
[199] vgl. http://www.duden.de/rechtschreibung/Protokoll#Bedeutung3a (22.01.2016)
[200] vgl. https://www.python.org/ (21.01.2016)
[201] vgl. http://www.duden.de/rechtschreibung/Quellcode (22.01.2016)
[202] vgl. http://www.duden.de/rechtschreibung/Netzwerk#b2-Bedeutung-3 (22.01.2016)
[203] vgl. u. a. http://pubs.opengroup.org/onlinepubs/009695399/basedefs/xbd_chap09.html (19.01.2016)

rekursives Herunterladen	bewirkt, dass alle Elemente die von der angegebenen *URL* ver*link*t werden, ebenfalls heruntergeladen werden[204]
Request	deutsch: *Anfrage*; wird in Form einer *Request-Message* (*Anfragenachricht*) vom *Client* zum *Server* ausgeschickt[205]
Request for Comments	*RFCs* sind eine Reihe technischer und organisatorischer Dokumente das Internet betreffend[206]
Request-Message	beschreibt *Request* des *Clients* an den *Server*; hält sich an bestimmtes Schema[207] (siehe Kapitel 2.2.3)
Request-URL	siehe *Request*; siehe *URL*
Response	deutsch: *Antwort*; wird in Form einer *Response-Message* (*Antwortnachricht*) vom *Server* an den *Client* geschickt[208]
Response-Message	deutsch: *Antwortnachricht*; beschreibt *Antwort* des *Servers* an den *Client*; hält sich an bestimmtes Schema[209] (siehe Kapitel 2.2.4)
Safari	*Browser* von Apple Inc.[210]
sed	(Stream Editor); Programm des GNU-Projekts, mit dem Text-*Datenströme* editiert werden[211]
Selenium	Ansammlung von *Tools* zur Automatisierung von *Browsern*[212]
Selenium IDE	*Tool* aus der Ansammlung der *Selenium-Tools*[213]
Selenium WebDriver	*Tool* aus der Ansammlung der *Selenium-Tools*[214]
Server	hier: wartet auf Verbindungs*anfragen* von *Clients*; antwortet auf diese durch Liefern der gewünschten Ressource oder Statusinformation[215]
serverseitig generierte dynamische Webseite	*Webseite*, die beim Abruf vom Server generiert wird; ausgeführter *Quelltext* ist für den *Client* nicht sichtbar; *Client* erhält lediglich die Ausgabe dieser Ausführung (Teil der im Rahmen dieser Arbeit getroffenen Klassifizierung von *Webseiten*)
Session	deutsch: *Sitzung*; Dialog, welcher sich über mehrere Zyklen von *Request* und *Response* erstreckt; innerhalb einer *Session* soll Rückbezug auf vergangene Kommunikation möglich sein[216]
Simple Mail Transfer Protocol	*Protokoll*, gemäß welchem E-Mails im Internet übertragen werden[217]
Sitzung	hier: siehe *Session*
Skript	hier: Liste von *Bash-Befehlen*, die der Reihe nach ausgeführt werden sollen; kleines Computerprogramm[218]

[204] vgl. https://www.gnu.org/software/wget/manual/wget.html#Recursive-Download (22.01.2016)
[205] vgl. http://www.w3.org/Protocols/rfc2616/rfc2616-sec5.html (19.01.2016)
[206] vgl. http://www.rfc-editor.org/ (22.01.2016)
[207] vgl. http://www.w3.org/Protocols/rfc2616/rfc2616-sec5.html (19.01.2016)
[208] vgl. http://www.w3.org/Protocols/rfc2616/rfc2616-sec6.html (19.01.2016)
[209] vgl. http://www.w3.org/Protocols/rfc2616/rfc2616-sec6.html (19.01.2016)
[210] vgl. http://www.apple.com/at/safari/ (19.01.2016)
[211] vgl. https://www.gnu.org/software/sed/ (19.01.2016)
[212] vgl. http://docs.seleniumhq.org/ (19.01.2016)
[213] vgl. http://docs.seleniumhq.org/docs/02_selenium_ide.jsp (19.01.2016)
[214] vgl. http://docs.seleniumhq.org/docs/02_selenium_ide.jsp (19.01.2016)
[215] vgl. http://tools.ietf.org/html/rfc2616#section-1.3 (19.01.2016)
[216] vgl. http://www.uni-weimar.de/medien/webis/teaching/lecturenotes/web-technology/unit-de-network-protocol3.pdf, Seite 40 (19.01.2016)
[217] vgl. https://tools.ietf.org/html/rfc5321 (19.01.2016)
[218] vgl. u. a. http://www.duden.de/rechtschreibung/Skript#Bedeutung4 (22.01.2016)

Slash	deutsch: Schrägstrich; von rechts oben nach links unten verlaufender Strich; Zeichen: „/“[219] (*ASCII* Code 47)
sort	Programm aus der Ansammlung der *GNU Coreutils* zur zeilenweisen Sortierung von Dateien[220]
Standardausgabe	Programm kann über diese Daten ausgeben; normalerweise mit dem Bildschirm verbunden.[221]
statische Webseite	*Webseite*, die statisch ist und nicht beim Abruf generiert wird (Teil der im Rahmen dieser Arbeit getroffenen Klassifizierung von *Webseiten*)
syntaktische Analyse	siehe *grammatische Analyse*
Syntax	hier: korrekte Verknüpfung sprachlicher Einheiten[222]
Tabulator	Durch Drücken der Tabulatortaste [↹] eingefügtes Zeichen[223] (*ASCII* Code 9)
Tag	Markierungselement von Beschreibungssprachen (hier: *HTML*) zur Strukturierung der Dokumente[224]
tail	Programm, das die letzten zehn (oder eine andere Anzahl an) Zeilen einer Textdatei auf dem Bildschirm ausgibt[225]
Terminal	siehe *Kommandozeile*
Token	Folge von logisch zusammengehörigen Einheiten (beispielsweise Folge von Zeichen)[226]
Tool	deutsch: *Werkzeug*; Computerprogramm (in der Regel von geringerem Umfang), das zusätzliche Aufgaben für ein bestimmtes Betriebssystem übernimmt[227]
Top-Level-Typ	übergeordneter Bestandteil eines *MIME-Types*;[228] Beispiele: text, image, audio, application
tr	Programm aus der Ansammlung der *GNU Coreutils* zum Ersetzen oder Löschen von Zeichen einer Eingabe[229]
Ubuntu Linux 14.04.3 LTS	Auf dem Linux-Kernel (Linux-Kern) basierendes Betriebssystem von Canonical Ltd. aus dem Jahr 2014[230]
Uniform Resource Locator	Schema für die Angabe einer *Internetadresse*; gibt an, welche Daten von welchem Computer unter Verwendung des angegebenen *Protokolls* übertragen werden sollen[231]
Untertyp	untergeordneter Bestandteil eines *MIME-Types*;[232] Beispiele für *Untertypen* des *Top-Level-Typs* „image“: gif, jpeg

[219] vgl. http://www.duden.de/rechtschreibung/Schraegstrich#b2-Bedeutung-b (22.01.2016)
[220] vgl. https://www.gnu.org/software/coreutils/manual/coreutils.html#sort-invocation (19.01.2016)
[221] vgl. http://www.linfo.org/standard_output.html (29.01.2016)
[222] vgl. http://www.duden.de/rechtschreibung/Syntax (22.01.2016)
[223] vgl. u. a. http://www-01.ibm.com/software/globalization/topics/keyboards/physical.html
[224] vgl. http://www.duden.de/rechtschreibung/Tag_Strukturelement_Markierung (22.01.2016)
[225] vgl. https://www.gnu.org/software/coreutils/manual/coreutils.html#tail-invocation (19.01.2016)
[226] vgl. http://www.duden.de/rechtschreibung/Token (22.01.2016) und https://www.ibm.com/developerworks/community/blogs/nlp/entry/tokenization?lang=en (22.01.2016)
[227] vgl. http://www.duden.de/rechtschreibung/Tool (22.01.2016)
[228] vgl. http://www.iana.org/assignments/media-types/media-types.xhtml (19.01.2016)
[229] vgl. https://www.gnu.org/software/coreutils/manual/coreutils.html#tr-invocation (19.01.2016)
[230] vgl. http://releases.ubuntu.com/14.04/ (21.01.2016)
[231] vgl. https://tools.ietf.org/html/rfc1738 (19.01.2016)
[232] vgl. http://www.iana.org/assignments/media-types/media-types.xhtml (19.01.2016)

URL Rewriting	Technik zur Codierung von *Session*-Information; *Session*-Information wird hierbei in *URL* codiert[233]
URL-Syntax	siehe *URL*; siehe *Syntax*
UTF-8	genormter Code für Ziffern, Buchstaben und Sonderzeichen[234]
Verzeichnis	Teil des Dateisystems, in dem Dateien abgelegt werden[235]
Webbrowser	siehe *Browser*
Webseite	Teil einer *Website*[236]
Webserver	siehe *Server*
Website	Gesamtheit der hinter einer *Internetadresse* stehenden *Webseiten*[237]
Werkzeug	hier: siehe *Tool*
`wget`	*Kommandozeilenprogramm* des GNU-Projekts zum Download von *Webseiten*[238]
Windows Internet Explorer	*Browser* der Microsoft Corporation[239]
World Wide Web Consortium	Konsortium zur Erstellung von Web-Standards[240]
Zeichenkette	Folge von Zeichen[241]
Zeichenfolge	siehe *Zeichenkette*
zustandslos	hier: alle *Anfragen* als voneinander unabhängig behandelnd[242]

[233] vgl. http://www.uni-weimar.de/medien/webis/teaching/lecturenotes/web-technology/unit-de-network-protocol3.pdf, Seite 42 (19.01.2016)
[234] vgl. https://tools.ietf.org/html/rfc3629 (19.01.2016)
[235] vgl. u. a. http://www.duden.de/rechtschreibung/Ordner#b2-Bedeutung-3 (22.01.2016)
[236] vgl. http://www.duden.de/rechtschreibung/Seite#b2-Bedeutung-11 (22.01.2016)
[237] vgl. http://www.duden.de/rechtschreibung/Website (22.01.2016)
[238] vgl. https://www.gnu.org/software/wget/ (19.01.2016)
[239] vgl. http://windows.microsoft.com/de-at/internet-explorer/download-ie (19.01.2016)
[240] vgl. http://www.w3.org/ (19.01.2016)
[241] vgl. http://www.duden.de/rechtschreibung/Zeichenkette (22.01.2016)
[242] vgl. https://tools.ietf.org/html/rfc2616, Abstract (19.01.2016)

C Literaturverzeichnis

Legende:

Fußnotennummer URL (Datum des letzten Zugriffs)

1 https://www.gnu.org/software/bash/manual/bashref.html (19.01.2016)

2 https://www.python.org/ (21.01.2016)

3 http://windows.microsoft.com/de-de/windows/windows-help#windows=windows-xp (21.01.2016)

4 http://releases.ubuntu.com/14.04/ (21.01.2016)

5 http://www.w3.org/ (19.01.2016)

6 https://www.ietf.org/ (19.01.2016)

7 https://www.mozilla.org/de/firefox/new/ (19.01.2016)

8 https://www.google.de/chrome/browser/desktop/index.html (19.01.2016)

9 http://windows.microsoft.com/de-at/internet-explorer/download-ie (19.01.2016)

10 https://www.microsoft.com/de-at/windows/microsoft-edge (19.01.2016)

11 http://www.apple.com/at/safari/ (19.01.2016)

12 http://www.opera.com/de (19.01.2016)

13 https://www.google.com (19.01.2016)

14 https://tools.ietf.org/html/rfc1738 (19.01.2016)

15 https://tools.ietf.org/html/rfc1738#section-2 (19.01.2016)

16 https://tools.ietf.org/html/rfc1738#section-2.1 (19.01.2016)

17 https://tools.ietf.org/html/rfc354 (19.01.2016)

18 https://tools.ietf.org/html/rfc5321 (19.01.2016)

19 https://tools.ietf.org/html/rfc1939 (19.01.2016)

20 https://tools.ietf.org/html/rfc3501 (19.01.2016)

21 https://tools.ietf.org/html/rfc4511 (19.01.2016)

22 https://tools.ietf.org/html/rfc5905 (19.01.2016)

23 https://tools.ietf.org/html/rfc2616, Abstract (19.01.2016)

24 http://tools.ietf.org/html/rfc2616#section-1.3 (19.01.2016)

25 https://www.rfc-editor.org/rfc/rfc793.txt (19.01.2016)

26 http://www.w3.org/Protocols/rfc2616/rfc2616-sec5.html (19.01.2016)

27 http://tools.ietf.org/html/rfc2616#section-5.1.1 (19.01.2016)

28 http://tools.ietf.org/html/rfc3986#section-1.1.3 (19.01.2016)

29 http://tools.ietf.org/html/rfc2616#section-9.3 (19.01.2016)

30 http://tools.ietf.org/html/rfc2616#section-9.5 (19.01.2016)

31 http://www.w3.org/Protocols/rfc2616/rfc2616-sec6.html (19.01.2016)

32 http://www.w3.org/Protocols/rfc2616/rfc2616-sec6.html (19.01.2016)

33 http://www.iana.org/assignments/http-status-codes/http-status-codes.xhtml (19.01.2016)

34 http://www.iana.org/assignments/http-status-codes/http-status-codes.xhtml (19.01.2016)

35 https://tools.ietf.org/html/rfc2818 (19.01.2016)

36 https://tools.ietf.org/html/rfc2818#section-1 (19.01.2016)

37 http://tools.ietf.org/html/rfc7230#section-2.3 (19.01.2016)

38 http://www.uni-weimar.de/medien/webis/teaching/lecturenotes/web-technology/unit-de-network-protocol3.pdf, Seite 40 (19.01.2016)

39 http://www.uni-weimar.de/medien/webis/teaching/lecturenotes/web-technology/unit-de-network-protocol3.pdf, Seite 42 (19.01.2016)

40 https://tools.ietf.org/html/rfc6265 (19.01.2016)

41 http://www.w3.org/2001/tag/2002/0129-mime (19.01.2016)

42 http://www.iana.org/assignments/media-types/media-types.xhtml (19.01.2016)

43 http://www.w3.org/standards/webdesign/htmlcss#whathtml (19.01.2016)

44 http://www.w3.org/TR/html401/struct/global.html#h-7.1 (19.01.2016)

45 http://www.w3.org/TR/html401/sgml/dtd.html (19.01.2016)

46 http://www.w3.org/standards/webdesign/htmlcss#whatcss (19.01.2016)

47 http://www.w3.org/TR/CSS1/#abstract (19.01.2016)

48 http://www.w3.org/TR/CSS21/intro.html#html-tutorial (19.01.2016)

49 u. a. https://support.mozilla.org/de/kb/Einstellungen-Fenster--Inhalts-Abschnitt (19.01.2016)

50 u. a. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference (19.01.2016)

51 http://ecma-international.org/ecma-262/5.1/ (19.01.2016)

52 https://products.office.com/de-at/excel (19.01.2016)

53 http://lynx.invisible-island.net/ (19.01.2016)

54 http://redmonk.com/sogrady/2013/07/25/language-rankings-6-13/ (19.01.2016)

55 http://php.net/ (19.01.2016)

56 http://www.asp.net/ (19.01.2016)

57 http://www.oracle.com/technetwork/java/javaee/jsp/index.html (19.01.2016)

58 http://php.net/manual/de/intro-whatis.php (19.01.2016)

59 http://redmonk.com/sogrady/2013/07/25/language-rankings-6-13/ (19.01.2016)

60 http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf (19.01.2016) und https://developer.mozilla.org/en-US/docs/Web/JavaScript (19.01.2016)

61 https://developer.mozilla.org/en-US/docs/Web/JavaScript (19.01.2016)

62 http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf (19.01.2016)

63 https://www.gnu.org/software/wget/ (19.01.2016)

64 http://www.gnu.org/licenses/gpl-3.0.de.html (19.01.2016)

65 u. a. http://csapp.cs.cmu.edu/3e/docs/chistory.html (19.01.2016)

66 u. a. http://pubs.opengroup.org/onlinepubs/9699919799/utilities/crontab.html (19.01.2016)
67 https://www.gnu.org/software/wget/manual/wget.html#Overview (19.01.2016)
68 https://www.gnu.org/software/wget/manual/wget.html#Invoking (19.01.2016)
69 https://www.gnu.org/software/wget/manual/wget.html#Logging-and-Input-File-Options (19.01.2016)
70 https://www.gnu.org/software/wget/manual/wget.html#Basic-Startup-Options (19.01.2016)
71 https://www.gnu.org/software/wget/manual/wget.html#Logging-and-Input-File-Options (19.01.2016)
72 https://www.gnu.org/software/wget/manual/wget.html#Download-Options (19.01.2016)
73 https://www.gnu.org/software/wget/manual/wget.html#Logging-and-Input-File-Options (19.01.2016)
74 https://www.gnu.org/software/wget/manual/wget.html#Download-Options (19.01.2016)
75 https://www.gnu.org/software/wget/manual/wget.html#Download-Options (19.01.2016)
76 https://www.gnu.org/software/wget/manual/wget.html#Directory-Options (19.01.2016)
77 https://www.gnu.org/software/wget/manual/wget.html#Logging-and-Input-File-Options (19.01.2016)
78 https://www.gnu.org/software/wget/manual/wget.html#HTTP-Options (19.01.2016)
79 https://www.gnu.org/software/wget/manual/wget.html#Recursive-Accept_002fReject-Options (19.02.2016)
80 https://www.gnu.org/software/wget/manual/wget.html#Recursive-Retrieval-Options (19.01.2016)
81 https://www.gnu.org/software/wget/manual/wget.html#Recursive-Retrieval-Options (19.01.2016)
82 https://www.gnu.org/software/wget/manual/wget.html#Recursive-Retrieval-Options (19.02.2016)
83 http://curl.haxx.se/ (19.01.2016)
84 http://curl.haxx.se/, curl is an open source command line tool and library (19.01.2016)
85 http://curl.haxx.se/docs/manpage.html, SYNOPSIS (19.01.2016)
86 http://curl.haxx.se/docs/manpage.html, OPTIONS (19.01.2016)
87 http://docs.seleniumhq.org/ (19.01.2016)
88 http://docs.seleniumhq.org/docs/02_selenium_ide.jsp (19.01.2016)
89 http://www.seleniumhq.org/docs/ (19.01.2016)
90 http://docs.seleniumhq.org/docs/02_selenium_ide.jsp, IDE Features, Menu Bar (19.01.2016)
91 http://docs.seleniumhq.org/docs/02_selenium_ide.jsp, IDE Features, Toolbar (19.01.2016)
92 http://docs.seleniumhq.org/docs/02_selenium_ide.jsp, IDE Features, Test Case Pane (19.01.2016)
93 http://release.seleniumhq.org/selenium-core/1.0.1/reference.html (19.01.2016)
94 http://www.gnu.org/software/coreutils/coreutils.html (22.01.2016)

95 https://www.gnu.org/software/coreutils/manual/coreutils.html#cat-invocation (19.01.2016)

96 https://www.gnu.org/software/coreutils/manual/coreutils.html#cut-invocation (19.01.2016)

97 http://www.gnu.org/software/grep/manual/grep.html (19.01.2016)

98 https://www.gnu.org/software/coreutils/manual/coreutils.html#head-invocation (19.01.2016)

99 https://www.gnu.org/software/coreutils/manual/coreutils.html#tail-invocation (19.01.2016)

100 https://www.gnu.org/software/sed/ (19.01.2016)

101 http://www.its.bldrdoc.gov/fs-1037/dir-010/_1451.htm (19.01.2016)

102 https://www.gnu.org/software/sed/manual/sed.html#Introduction (19.01.2016)

103 https://www.gnu.org/software/sed/manual/sed.html#Invoking-sed (19.01.2016)

104 https://www.gnu.org/software/sed/manual/sed.html#index-Global-substitution-108 (19.01.2016)

105 https://www.gnu.org/software/coreutils/manual/coreutils.html#sort-invocation (19.01.2016)

106 https://www.gnu.org/software/coreutils/manual/coreutils.html#tr-invocation (19.01.2016)

107 http://www.iue.tuwien.ac.at/phd/demel/node107.html (19.01.2016)

108 https://www.python.org/ (19.01.2016)

109 https://docs.python.org/2/library/htmlparser.html (19.01.2016)

110 https://www.gnu.org/software/bison/ (19.01.2016)

111 http://flex.sourceforge.net/ (19.01.2016)

112 https://products.office.com/de-at/excel (19.01.2016)

113 https://tools.ietf.org/html/rfc4180#page-2 (19.01.2016)

114 https://www.python.org/ (19.01.2016)

115 https://www.python.org/dev/peps/pep-0008/#inline-comments (19.01.2016)

116 https://tools.ietf.org/html/rfc3629 (19.01.2016)

117 https://docs.python.org/2/library/codecs.html (19.01.2016)

118 https://docs.python.org/2/tutorial/inputoutput.html (19.01.2016)

119 http://www.seleniumhq.org/projects/webdriver/ (19.01.2016)

120 https://docs.python.org/2/tutorial/inputoutput.html (19.01.2016)

121 u. a. http://pubs.opengroup.org/onlinepubs/009695399/basedefs/xbd_chap09.html (19.01.2016)

122 https://www.gnu.org/software/sed/manual/html_node/Regular-Expressions.html (20.01.2016)

123 http://www.asp.net/ (19.01.2016)

124 http://www.duden.de/rechtschreibung/Add_on (19.01.2016)

125 u. a. https://support.mozilla.org/de/kb/autovervollstaendigung-adressleiste (19.01.2016)

126 http://tools.ietf.org/html/rfc20 (22.01.2016)

127 u. a. https://www.dict.cc/?s=Arbeitsverzeichnis (22.01.2016)

128 http://tools.ietf.org/html/rfc20 (22.01.2016)

129 https://www.gnu.org/software/bash/ (19.01.2016)

130 http://www.duden.de/rechtschreibung/Befehl#Bedeutung1b (22.01.2016)

131 https://www.gnu.org/software/bison/ (19.01.2016)
132 http://www.duden.de/rechtschreibung/Browser (22.01.2016)
133 u. a. http://csapp.cs.cmu.edu/3e/docs/chistory.html (19.01.2016)
134 http://www.w3.org/standards/webdesign/htmlcss#whatcss (19.01.2016)
135 https://www.gnu.org/software/coreutils/manual/coreutils.html#cat-invocation (19.01.2016)
136 http://tools.ietf.org/html/rfc2616#section-1.3 (19.01.2016)
137 https://tools.ietf.org/html/rfc4180#page-2 (19.01.2016)
138 https://tools.ietf.org/html/rfc6265 (19.01.2016) und
http://www.duden.de/rechtschreibung/Cookie (22.01.2016)
139 u. a. http://pubs.opengroup.org/onlinepubs/9699919799/utilities/crontab.html (19.01.2016)
140 http://curl.haxx.se/ (19.01.2016)
141 https://www.gnu.org/software/coreutils/manual/coreutils.html#cut-invocation (19.01.2016)
142 http://www.linfo.org/path.html (22.01.2016)
143 http://www.its.bldrdoc.gov/fs-1037/dir-010/_1451.htm (19.01.2016)
144 http://www.its.bldrdoc.gov/fs-1037/dir-011/_1544.htm (22.01.2016)
145 http://www.duden.de/rechtschreibung/Dokumenttyp (22.01.2016)
146 http://www.ecma-international.org/ (22.01.2016)
147 http://ecma-international.org/ecma-262/5.1/ (19.01.2016)
148 http://www.iana.org/assignments/media-types/media-types.xhtml (19.01.2016)
149 https://tools.ietf.org/html/rfc354 (19.01.2016)
150 http://flex.sourceforge.net/ (19.01.2016)
151 http://tools.ietf.org/html/rfc2616#section-9.3 (19.01.2016)
152 https://www.w3.org/Graphics/GIF/spec-gif89a.txt (23.01.2016)
153 http://www.gnu.org/software/coreutils/coreutils.html (22.01.2016)
154 u. a. http://www.gnu.org/licenses/gpl-3.0.html
155 https://www.google.de/chrome/browser/desktop/index.html (19.01.2016)
156 https://www.google.com (19.01.2016)
157 http://www.iue.tuwien.ac.at/phd/demel/node107.html (19.01.2016)
158 http://www.gnu.org/software/grep/manual/grep.html (19.01.2016)
159 https://www.gnu.org/software/coreutils/manual/coreutils.html#head-invocation (19.01.2016)
160 http://www.uni-weimar.de/medien/webis/teaching/lecturenotes/web-technology/unit-de-network-protocol3.pdf, Seite 42 (19.01.2016)
161 http://www.duden.de/rechtschreibung/Homepage (22.01.2016)
162 https://docs.python.org/2/library/htmlparser.html (19.01.2016)
163 https://tools.ietf.org/html/rfc2818 (19.01.2016)
164 http://www.duden.de/rechtschreibung/Hypertext (22.01.2016)
165 http://www.w3.org/standards/webdesign/htmlcss#whathtml (19.01.2016)
166 https://tools.ietf.org/html/rfc2616, Abstract (19.01.2016)

167 https://www.ietf.org/ (19.01.2016)

168 https://tools.ietf.org/html/rfc3501 (19.01.2016)

169 http://www.duden.de/rechtschreibung/Internetadresse (22.01.2016)

170 http://www.duden.de/rechtschreibung/Internetsurfen (22.01.2016)

171 http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf (19.01.2016) und https://developer.mozilla.org/en-US/docs/Web/JavaScript (19.01.2016)

172 http://www.oracle.com/technetwork/java/javaee/jsp/index.html (19.01.2016)

173 http://jpeg.org/ (23.01.2016)

174 http://www.duden.de/rechtschreibung/Kommandozeile (22.01.2016)

175 http://www.duden.de/rechtschreibung/Kommandozeile (22.01.2016)

176 u. a. https://www.ibm.com/developerworks/community/blogs/nlp/entry/tokenization?lang=en (22.01.2016)

177 https://tools.ietf.org/html/rfc4511 (19.01.2016)

178 http://www.duden.de/rechtschreibung/Link (22.01.2016)

179 http://www.duden.de/rechtschreibung/Log_in (22.01.2016)

180 http://www.duden.de/rechtschreibung/Log_in (22.01.2016)

181 http://lynx.invisible-island.net/ (19.01.2016)

182 https://wiki.selfhtml.org/wiki/HTML/Kopfdaten/meta (22.01.2016)

183 https://www.microsoft.com/de-at/windows/microsoft-edge (19.01.2016)

184 https://products.office.com/de-at/excel (19.01.2016)

185 http://windows.microsoft.com/de-de/windows/windows-help#windows=windows-xp (21.01.2016)

186 http://www.w3.org/2001/tag/2002/0129-mime (19.01.2016)

187 https://www.mozilla.org/de/firefox/new/ (19.01.2016)

188 https://tools.ietf.org/html/rfc5905 (19.01.2016)

189 http://www.duden.de/rechtschreibung/Open_Source_Software (22.01.2016)

190 http://www.opera.com/de (19.01.2016)

191 u. a. http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2005/n1905.pdf, Seite 16, Kap. 1.3.1 argument (22.01.2016) und http://www.w3.org/standards/webdesign/htmlcss#whathtml (19.01.2016)

192 https://www2.informatik.uni-erlangen.de/EN/teaching/SS2012/HalloWelt/PA_2012.pdf, Seite 3 (22.01.2016)

193 http://php.net/ (19.01.2016)

194 u. a. http://www.linfo.org/pipe.html (22.01.2016)

195 https://tools.ietf.org/html/rfc2046#section-4.1.3 (19.01.2016)

196 http://tools.ietf.org/html/rfc2616#section-9.5 (19.01.2016)

197 https://tools.ietf.org/html/rfc1939 (19.01.2016)

198 http://www.adobe.com/devnet/postscript.html (23.01.2016)

199 http://www.duden.de/rechtschreibung/Protokoll#Bedeutung3a (22.01.2016)

200 https://www.python.org/ (21.01.2016)

201 http://www.duden.de/rechtschreibung/Quellcode (22.01.2016)

202 http://www.duden.de/rechtschreibung/Netzwerk#b2-Bedeutung-3 (22.01.2016)

203 u. a. http://pubs.opengroup.org/onlinepubs/009695399/basedefs/xbd_chap09.html (19.01.2016)

204 https://www.gnu.org/software/wget/manual/wget.html#Recursive-Download (22.01.2016)

205 http://www.w3.org/Protocols/rfc2616/rfc2616-sec5.html (19.01.2016)

206 http://www.rfc-editor.org/ (22.01.2016)

207 http://www.w3.org/Protocols/rfc2616/rfc2616-sec5.html (19.01.2016)

208 http://www.w3.org/Protocols/rfc2616/rfc2616-sec6.html (19.01.2016)

209 http://www.w3.org/Protocols/rfc2616/rfc2616-sec6.html (19.01.2016)

210 http://www.apple.com/at/safari/ (19.01.2016)

211 https://www.gnu.org/software/sed/ (19.01.2016)

212 http://docs.seleniumhq.org/ (19.01.2016)

213 http://docs.seleniumhq.org/docs/02_selenium_ide.jsp (19.01.2016)

214 http://docs.seleniumhq.org/docs/02_selenium_ide.jsp (19.01.2016)

215 http://tools.ietf.org/html/rfc2616#section-1.3 (19.01.2016)

216 http://www.uni-weimar.de/medien/webis/teaching/lecturenotes/web-technology/unit-de-network-protocol3.pdf, Seite 40 (19.01.2016)

217 https://tools.ietf.org/html/rfc5321 (19.01.2016)

218 u. a. http://www.duden.de/rechtschreibung/Skript#Bedeutung4 (22.01.2016)

219 http://www.duden.de/rechtschreibung/Schraegstrich#b2-Bedeutung-b (22.01.2016)

220 https://www.gnu.org/software/coreutils/manual/coreutils.html#sort-invocation (19.01.2016)

221 http://www.linfo.org/standard_output.html (29.01.2016)

222 http://www.duden.de/rechtschreibung/Syntax (22.01.2016)

223 u. a. http://www-01.ibm.com/software/globalization/topics/keyboards/physical.html

224 http://www.duden.de/rechtschreibung/Tag_Strukturelement_Markierung (22.01.2016)

225 https://www.gnu.org/software/coreutils/manual/coreutils.html#tail-invocation (19.01.2016)

226 http://www.duden.de/rechtschreibung/Token (22.01.2016) und https://www.ibm.com/developerworks/community/blogs/nlp/entry/tokenization?lang=en (22.01.2016)

227 http://www.duden.de/rechtschreibung/Tool (22.01.2016)

228 http://www.iana.org/assignments/media-types/media-types.xhtml (19.01.2016)

229 https://www.gnu.org/software/coreutils/manual/coreutils.html#tr-invocation (19.01.2016)

230 http://releases.ubuntu.com/14.04/ (21.01.2016)

231 https://tools.ietf.org/html/rfc1738 (19.01.2016)

232 http://www.iana.org/assignments/media-types/media-types.xhtml (19.01.2016)

233 http://www.uni-weimar.de/medien/webis/teaching/lecturenotes/web-technology/unit-de-network-protocol3.pdf, Seite 42 (19.01.2016)

234 https://tools.ietf.org/html/rfc3629 (19.01.2016)

235 u. a. http://www.duden.de/rechtschreibung/Ordner#b2-Bedeutung-3 (22.01.2016)

236 http://www.duden.de/rechtschreibung/Seite#b2-Bedeutung-11 (22.01.2016)

237 http://www.duden.de/rechtschreibung/Website (22.01.2016)

238 https://www.gnu.org/software/wget/ (19.01.2016)

239 http://windows.microsoft.com/de-at/internet-explorer/download-ie (19.01.2016)

240 http://www.w3.org/ (19.01.2016)

241 http://www.duden.de/rechtschreibung/Zeichenkette (22.01.2016)

242 https://tools.ietf.org/html/rfc2616, Abstract (19.01.2016)

D Abbildungsverzeichnis

E Tabellenverzeichnis

F Quelltextverzeichnis

G Beiliegendes Speichermedium

Folgende Dateien sind auf dem beiliegenden Speichermedium vorhanden:

- `sprechstundenliste.html`
- `modulliste.html`
- `lynxdump.txt`