Stephan Czysch

So holen Sie sich Daten von Webseiten: **Scraping mit XPath & CSSPath**

Extrahieren Sie bisher über Markieren und Kopieren Daten aus Webseiten und Sie fragen sich, ob das nicht schneller, fehlerfreier oder gar automatisch geht? Die Antwort sollte Sie nicht verwundern: Ja, klar geht das! Und es ist deutlich einfacher, als Sie denken! Denn mit der richtigen Referenz auf die gewünschten Daten lassen sich diese im Handumdrehen extrahieren. Oder anders ausgedrückt: scrapen.
Das klingt nach Zauberei? Das ist es beileibe nicht! Erfahren Sie, wie Sie durch den Einsatz von XPath und CSSPath und passender Tools deutlich effizienter Daten auslesen können.

DER AUTOR

Stephan Czysch ist Geschäftsführer des Online-Marketing-Teams von Dept (*deptagency.com*) in Berlin. Seine Leidenschaft gilt besseren Websites. Stephan spricht regelmäßig auf Konferenzen zu Themen wie Online-Marketing-Strategien und datengetriebenes SEO. Sein Wissen können Sie in Form seiner Bücher konsumieren – oder besuchen Sie mal eins seiner Tagesseminare.

Ich hoffe sehr, dass Sie nicht direkt beschämt die Website Boosting zur Seite legen, wenn Sie den folgenden Begriff nochmals lesen: **Scraping**. Das klingt irgendwie böse und nach einer Grauzone des Internets, meinen Sie nicht auch? Der Begriff Scraping, vereinzelt auch Web Scraping genannt, beschreibt den Prozess, durch den Inhalte aus Webseiten extrahiert werden können.

Speziell für Suchmaschinenoptimierer ist die Arbeit mit gescrapten Daten an der Tagesordnung. Denn woher soll ein SEO-Tool z. B. wissen, auf welcher Position eine Website in der Google-Suche gefunden wird, ohne diese Daten aus den Suchergebnissen herauszulesen? Oder woher kann ein Tool die aktuelle H1-Überschrift einer Website sowie die momentan hinterlegte Meta Description liefern, ohne dazu auf die abzufragende Seite und deren Quelltext zuzugreifen? Der Quelltext einer Webseite ist essenziell wichtig, wenn es um das **Scraping von Daten** geht.

Zu beachten ist dabei, dass zwischen dem gerenderten und dem nicht-gerenderten Quelltext merklich Unterschiede bestehen können. Deutlich wird dies bei Webseiten, bei denen durch den Einsatz von JavaScript der initial geladene Quelltext verändert oder erweitert wird.

Den Quelltext einer Webseite können Sie in Ihrem Browser durch einen Rechtsklick, gefolgt von „Seitenquelltext anzeigen“ (oder ähnlich) ansehen. Den gerenderten Quelltext sehen Sie, wenn Sie z. B. im Firefox- oder Chrome-Browser nach dem Rechtsklick den Punkt „Objekt untersuchen“ (oder ähnlich) anwählen. In beiden Browsern führt ein Druck auf F12 zum jeweiligen **Inspector-Tool**.

An dieser Stelle ein kleiner Tipp: Für den Chrome-Browser gibt es ein fantastisches Plug-in, das Unterschiede zwischen den beiden Quelltextversionen sichtbar macht: **View Rendered Source**.

Das Plug-in finden Sie z. B. unter *http://einfach.st/stefanczysch* (zusammen mit weiteren hilfreichen Plug-ins sowie sogenannten Bookmarklets) oder direkt in den Chrome-Erweiterungen.

Ein Beispiel für eine Website, die für Nutzer und Suchmaschinen gleichermaßen relevante Seiteninhalte per JavaScript verändert, ist der Online-Shop von Conrad Electronic. Beim Aufrufen des nicht-gerenderten Quelltexts einer Kategorieseite ist als Seitentitel ‹title›%category-title% günstig online kaufen bei Conrad‹/title› hinterlegt – nicht sonderlich aussagekräftig, oder? Durch JavaScript wird der Platzhalter %category-title% durch den richtigen Inhalt ersetzt.

Was Sie über das Document Object Model (DOM) wissen müssen

Nach dem erfolgreichen Abruf liegt die angefragte Webseite zunächst als mittels HTML formatiertes Textdokument im Browser vor. In diesem Dokument sind einzelne Elemente ineinander verschachtelt. So enthält beispielsweise der ‹head›-Bereich des Dokuments den Seiten-

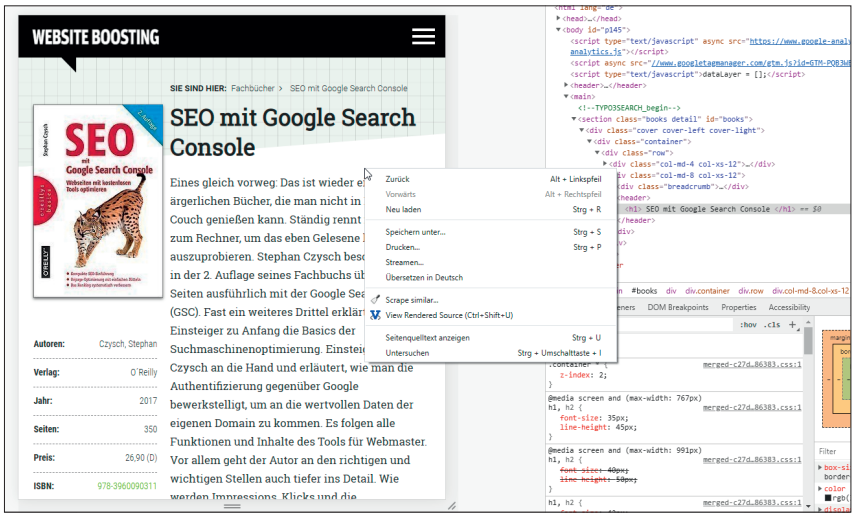

Abb.1: Durch einen Rechtsklick können Sie das jeweilige Inspector-Tool von Chrome oder Firefox zur Analyse des gerenderten Quelltexts aufrufen – oder alternativ per Druck auf die F12-Taste

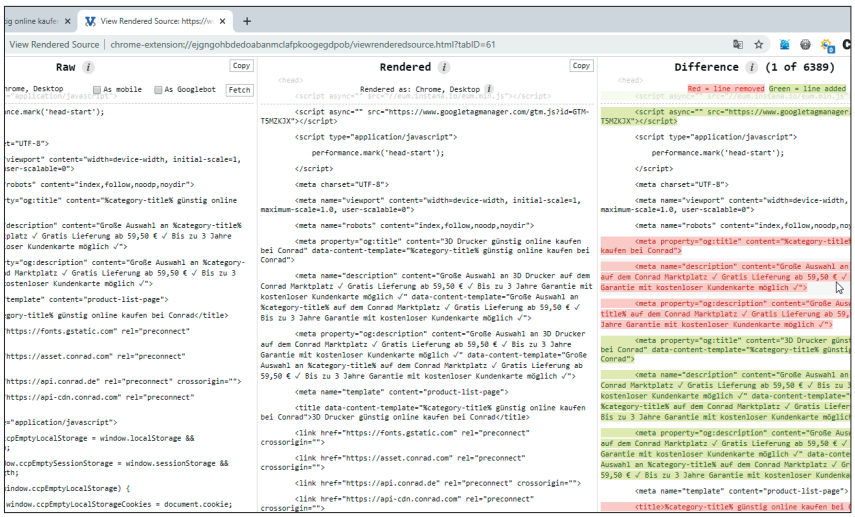

Abb.2: Über das Chrome-Browser-Plug-in „View Rendered Source“ können Sie den gerenderten mit dem initialen Quelltext vergleichen

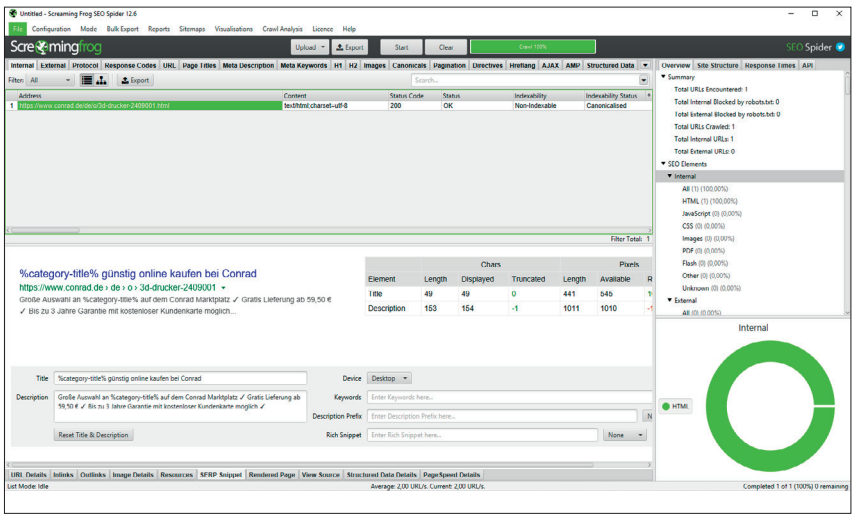

Abb.3: Ohne die Ausführung von JavaScript sind unter anderem Seitentitel und Meta Description im Online-Shop von Conrad Electronic nicht gut optimiert

WICHTIGER HINWEIS

Dieser Beitrag bezieht sich darauf, wie Sie Daten aus Ihrer eigenen Website oder nach vorheriger Erlaubnis von Fremdwebsites auslesen können. Beachten Sie, dass das (automatische) Auslesen von Fremdwebsites ohne Einverständnis je nach den extrahierten Daten rechtliche Grenzen überschreiten kann. Kontaktieren Sie deshalb im Zweifelsfall einen Anwalt, sofern Sie Daten v. a. gewerblich ohne Erlaubnis aus Fremdwebsites auslesen möchten. Achten Sie zudem darauf, dass Sie nicht zu viele zu schnelle Anfragen an solche Websites stellen. Denn dies könnte im Extremfall zu einer Überlastung oder gar einem vollständigen Ausfall der Website führen und als Denial-of-Service-Angriff interpretiert werden.

titel (‹title›) oder im ‹body› ist eine Tabelle (‹table›) in einem ‹div›-Container enthalten.

Der Browser fängt mit der Datenübertragung an, diesen in HTML formatierten Text zu interpretieren (auch „parsen“ genannt, vom Englischen to parse, in etwa analysieren). Dabei wird der HTML-Text in seine Objektstruktur übersetzt – das sogenannte DOM (für Document Object Model). Die einzelnen mittels HTML formatierten Inhalte werden dabei in einer Baumstruktur dargestellt.

Im Wiki von selfhtml.org ist unter der Adresse *https://wiki.selfhtml.org/wiki/JavaScript/DOM* eine sehr anschauliche Darstellung des DOM zu finden (siehe Abb. 4). Dieses kann, wie am Beispiel von Conrad Electronic beschrieben, mittels JavaScript verändert werden. Dort wird der initial hinterlegte Seitentitel durch einen optimierten ersetzt.

Für das Auslesen von Seiteninhalten ist die DOM-Struktur sehr hilfreich, da sich einzelne Bereiche, die sogenannten Knoten (englisch: nodes) ansprechen lassen. Wie das geht? Mittels XPath- oder CSSPath-Angaben. Diese schauen wir uns nun an!

Einführung in XPath

Damit dieser Artikel anwendungsorientiert bleibt, möchte ich zu den gesamten technischen Hintergründen nicht zu sehr ins Detail gehen. Die erfolgreich in Ihrem Browser aufgerufene Webseite liegt in der in Abb. 4 dargestellten Baumstruktur vor und wurde vereinfacht gesagt in einen XML-Aufbau überführt. XML steht für **Extensible Markup Language** – ins Deutsche übersetzt: erweiterbare Auszeichnungssprache.

Das Schöne an XML-Dateien ist deren strukturierter Aufbau in Form der untereinander in Beziehung stehenden Knoten. Diese Struktur zeigt Abbildung 5 sehr schön: Die einzelnen Knoten, in diesem Fall einer XML-Sitemap, lassen sich öffnen und schließen. Durch die Einrückung sind zudem die Hierarchien sichtbar. So liegen innerhalb von <url>-Angaben einzelne Bilder (<image>), die wiederum zusätzliche Daten wie die Bild-Adresse enthalten.

Warum müssen Sie dies wissen? Weil mittels XPath auf diese Knoten zugegriffen werden kann. **XPath steht für XML Path Language** – durch die entsprechenden Anweisungen lässt sich der gewünschte Knoten ansprechen und sein Inhalt auslesen.

Führen Sie sich nochmals die Baumstruktur in Abb. 4 vor Augen, dann sehen Sie, dass die H1-Überschrift ein Knoten innerhalb von main ist. Doch die Beziehung lässt sich noch genauer beschreiben: Main ist innerhalb des Body-Knotens zu finden, der seinerseits unterhalb von HTML liegt. Entsprechend kann mittels *html › main › body › h1* die Überschrift

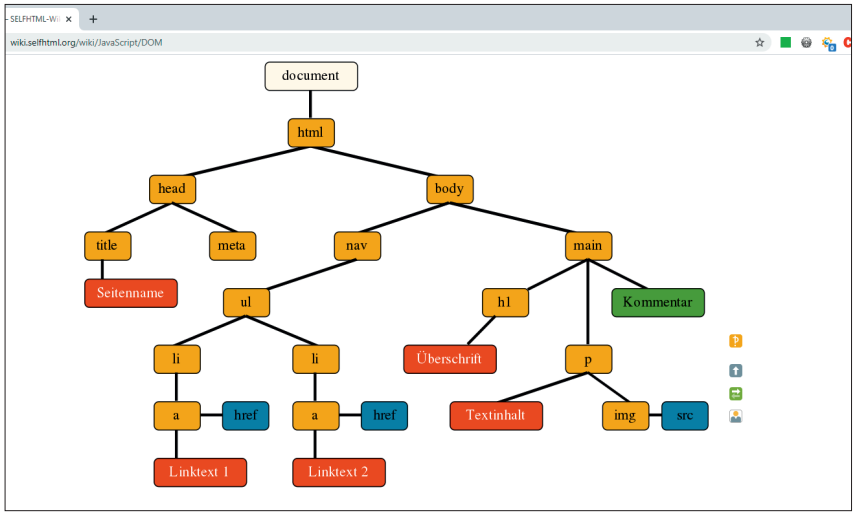

Abb.4: Im sehr lesenswerten Wiki von selfhmtl.org ist diese Darstellung des DOM zu finden

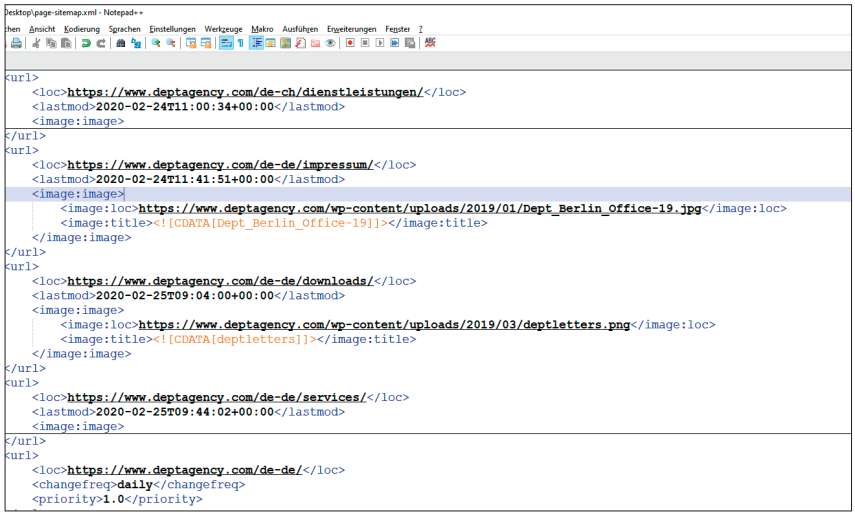

Abb.5: Der strukturierte hierarchische Aufbau dieser XML-Sitemap lässt sich hier gut sehen; einzelne Knoten können aus- und eingeklappt werden

angesprochen werden. In XPath übersetzt bedeutet dies */html/main/body/h1*. Diese Verschachtelungen sind im Quelltext sichtbar, da einzelne Elemente einander umschließen können und damit in Beziehung zueinander stehen.

Dankenswerterweise müssen Sie die einzelnen Verschachtelungen nicht selbst durchanalysieren und den XPath zur gewünschten Stelle runtertippen. Denn unter anderem erstellen die bereits genannten Browser Firefox und Chrome den XPath für Sie – den schon gezeigten Inspector-Tools sei Dank!

Am einfachsten gelangen Sie zum XPath, indem Sie mit der **Untersuchen-Funktion** arbeiten – diese können Sie z. B. durch einen Rechtsklick aktivieren. In Abb. 6 wurde der Breadcrumb-Pfad oberhalb des Buchtitels markiert. Durch einen Rechtsklick im Inspector kann unter Copy der XPath kopiert werden.

Je nach gewünschter Syntax (XPath oder Full XPath) wird folgende Referenz von Chrome geliefert:

» XPath: //*[@id="books"]/div[1]/div/div/div[2]/div
» Full XPath: /html/body/main/section/div[1]/div/div/div[2]/div

Beide beschreiben mit unterschiedlicher Syntax exakt dasselbe und beziehen sich dabei auf den Ausschnitt des Quelltexts in Abb. 7.

Schauen wir uns einmal die XPaths im Detail an, damit Sie verstehen, was hier passiert.

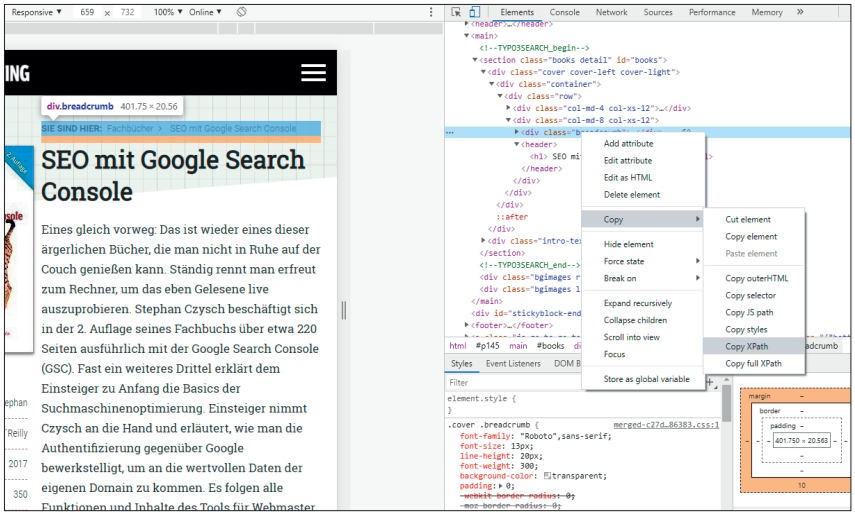

Abb.6: Dank beispielsweise des Chrome Inspectors können Sie den XPath zum gewünschten Abschnitt erstellen – in diesem Fall zum im linken Teil der Abbildung markierten Breadcrumb-Pfad

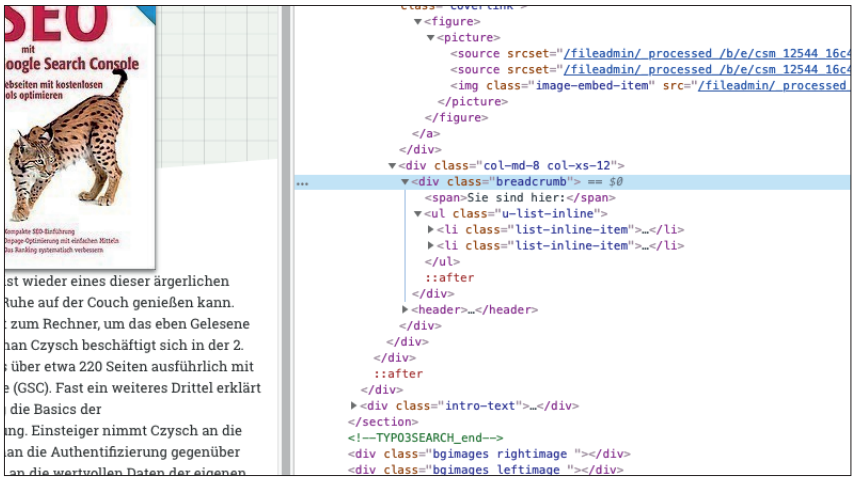

Abb. 7: Die Knotenstruktur ist im Chrome DevTools sehr schön zu sehen – ein Klick auf die kleinen Pfeilspitzen öffnet oder schließt einen Knoten

XPath: //*[@id="books"]/div[1]/div/div/div[2]/div

//*[@id="books"] ist als eine zusammenhängende Angabe zu sehen. Durch //* wird ein beliebiger Knoten unabhängig von seiner Position im Baum angesprochen, der als Attribut (wird durch das @ definiert) id="books" verwendet. Dieser Ausdruck wird von *‹section class="books detail" id="books"›* gematcht. Anstelle von section hätte auch jedes andere Element kommen können.

Die nun nachfolgenden Angaben beziehen sich auf nicht genauer benannte DIV-Container. Genauer spezifiziert werden könnten diese beispielsweise durch ihre Klassennamen (wie class="klassenname") oder gar eine ID (beispielsweise id="idname"). Hinter manchen DIV steht in eckiger Klammer eine Zahl. Diese Zahl definiert, auf welches gefundene Element sich bezogen wird, wenn auf derselben Hierarchiestufe mehrere dieser Elemente vorhanden sind. Klingt kompliziert? Dann schauen wir uns ein Beispiel an!

Innerhalb der ‹section› (siehe Abb. 7) befinden sich die beiden DIV-Container *‹div class="cover cover-left cover-light"›* sowie *‹div class="intro-text"›* auf derselben Ebene – sprich, sie sind nicht ineinander verschachtelt. Durch //*[@id="books"]/div[1]/ zeigt die Referenz auf das im Quelltext zuerst kommende DIV – in diesem Fall also *‹div class="cover cover-left cover-light"›*. Analog sind die weiteren Angaben zu lesen.

Full XPath: /html/body/main/section/div[1]/div/div/div[2]/div

Schauen wir noch auf die zweite Variante, die bei Chrome als Full XPath benannte Referenz. Diese ist mit dem bereits vermittelten Wissen einfacher nachzuvollziehen, da der Weg durch die Baumstruktur genau benannt wird. Als Pfad wird angegeben, dass innerhalb des Knotens ‹html zu ‹body und so weiter durchnavigiert werden soll. Die Ansprache der DIV-Container ist mit der ersten Variante identisch.

Die Frage ist jetzt, ob das nicht auch einfacher geht. Denn was ist, wenn sich der Quelltext verändert und z. B. noch ein weiterer DIV-Container im Quelltext vorkommt – oder gar eine Hierarchiestufe herausgelöscht wird? Dann würde der Zeiger ins Leere deuten.

Und es geht einfacher! Durch den Bezug auf z. B. Klassennamen oder gar IDs können XPaths kürzer und dennoch genauer beschrieben werden. Zur Information: In einem HTML-Dokument darf ein Klassenname mehrfach verwendet werden, eine ID allerdings nur einmal.

Schauen wir nochmals auf den Quelltext in Abb. 7. Wir wollen uns auf den Breadcrumb beziehen – dieser ist innerhalb eines DIV-Containers zu finden, der den Klassenamen class="breadcrumb" trägt. Durch den XPath //div[@class="breadcrumb"] könnten wir folglich die Referenz auf diesen Bereich setzen.

So viel zur Theorie! Wenn Sie mehr über die Syntax von XPath finden möchten, dann kann ich Ihnen das W3schools-Tutorial unter *www.w3schools.com/xml/xpath_syntax.asp* empfehlen.

Was Sie über CSSPath wissen müssen

Der Syntax von XPath ähnlich sind die **CSSPath-Selektoren**. Für das Styling von Websites werden CSS-Angaben eingesetzt. CSS steht dabei für Cascading Style Sheets, ins Deutsche in etwa

mit mehrstufige Formatvorlagen zu übersetzen. Mehrstufig deshalb, weil sich Styling-Angaben vererben oder weiter spezifiziert werden können, um z. B. nur einen einzelnen Absatz anders darzustellen.

Innerhalb von CSSPath kann ebenfalls über die Klassen- oder ID-Namen auf einzelne Knoten zugegriffen werden. Für das bereits bekannte Beispiel des Breadcrumb-Pfads ist *#books › div.cover.cover-left.cover-light › div › div › div.col-md-8.col-xs-12 › div* ein möglicher CSSPath.

Durch das vorangestellte Rautezeichen wird sich auf eine ID bezogen, durch die Punkte werden die Klassennamen bestimmt. Während bei XPath das x-te Element über den Wert in eckigen Klammern definiert wurde, gibt es bei CSSPath die Angabe nth-child (Nummer). Um den CSSPath eines Knotens zu erhalten, können Sie wiederum auf die Inspector-Tools von Firefox, Chrome und Co. zurückgreifen. In der aktuellen Chrome-Version ist der CSSPath als „Copy Selector" benannt.

Werfen Sie zur Vertiefung gerne einen Blick auf *www.w3schools.com/cssref/trysel.asp*.

Fortgeschrittene Möglichkeiten von CSSPath und XPath

Ob Sie CSSPath oder XPath verwenden, ist egal – CSSPath wird nachgesagt, dass die Suche im Vergleich etwas schneller abläuft. Bei normalen Anwendungen sollten Sie jedoch keinen Unterschied feststellen können.

Neben den hier als Beispiele verwendeten einfachen Selektoren können sowohl CSSPath als auch XPath deutlich verfeinert werden. Über Operatoren wie „contains" können z. B. nur bestimmte Elemente extrahiert werden. Anstatt beispielsweise jegliches iframe zu extrahieren, kann mittels des XPath //iframe[contains(@src ‚www.youtube.com/embed/')] definiert werden, dass nur iframes von Youtube-Videos extrahiert werden.

Weitere Beispiele gefällig?

» //div[@class="artikel"]/p[last()-1] sorgt dafür, dass der vorletzte Paragraf angesprochen wird.
» //a[starts-with(@href,'mailto')] extrahiert E-Mail-Adressen aus Webseiten.
» tr:nth-child(even) spricht mittels CSSPath geradzahlige Tabellenreihe (tr für tablerow) an, also 2,4,6, …
» //a[not(contains(@href, ‚google.de'))]/@href liefert alle Links, die nicht auf google.de zeigen.
» /descendant::h2[position() >= 0 and position() <= 3] extrahiert die ersten drei H2-Überschriften.

Mehr über Operatoren finden Sie unter *www.w3schools.com/cssref/css_selectors.asp* sowie *http://einfach.st/sikiself4*.

Mit welchen Tools können Sie Daten aus Webseiten scrapen?

Sie kennen jetzt die Syntax der beiden Extraktionswege – aber noch nicht die Tools, mit denen Sie die gewünschten Daten aus Webseiten extrahieren können. Wie nicht anders zu erwarten, gibt es eine ganze Reihe von Tools, die für die Datenextraktion eingesetzt werden können.

Web Scraping mit dem Screaming Frog

Wenn es darum geht, über viele verschiedene Adressen Inhalte zu extrahieren, dann ist ein Crawler wie der Screaming Frog das Tool der Wahl. Über die Crawl-Konfiguration können Sie im Unterpunkt Custom den Bereich Extraction finden.

Dort können Sie wählen, ob per XPath, CSSPath oder regulären Ausdruck (siehe dazu Website Boosting 60) aus dem Quelltext Daten extrahiert werden sollen. Durch den grünen Haken sehen Sie, ob die Syntax valide ist, also ob z. B. eine Klammer nicht richtig geschlossen ist.

Die über die hier definierten Angaben extrahierten Daten finden Sie anschließend im „Custom Extraction"-Bereich des Screaming Frog. Dieser ist in Abb. 8 ebenfalls zu sehen. Der Screaming Frog bietet unterschiedliche Extraktionsformen:

» Extract HTML Element: Extrahiert das Element mitsamt seinem HTML-Inhalt.
» Extract Inner HTML: Hierdurch wird der HTML-Inhalt des gewählten Elements gezogen. Sofern dieses ebenfalls HTML enthält, wird dies mit ausgegeben.
» Extract Text: Wenn Sie die HTML-Auszeichnungen nicht benötigen, dann

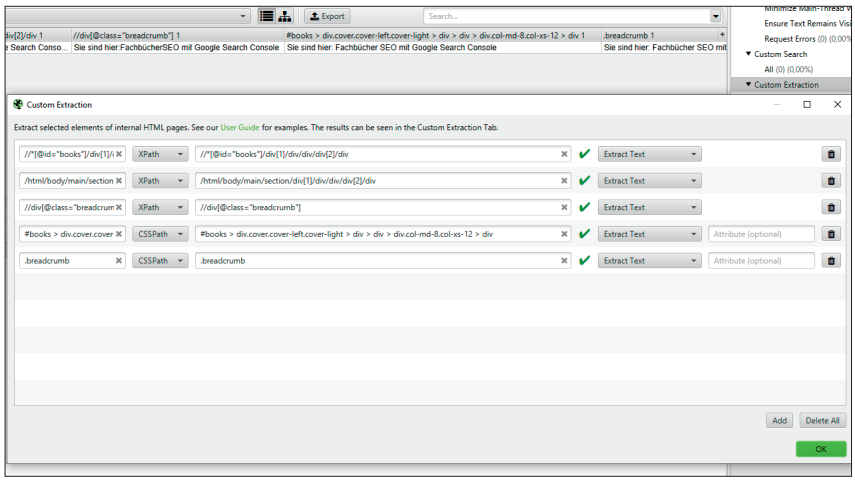

Abb. 8: Im Screaming Frog können Sie unter Configuration => Custom => Extraction definieren, was Sie wie aus der Website extrahieren möchten

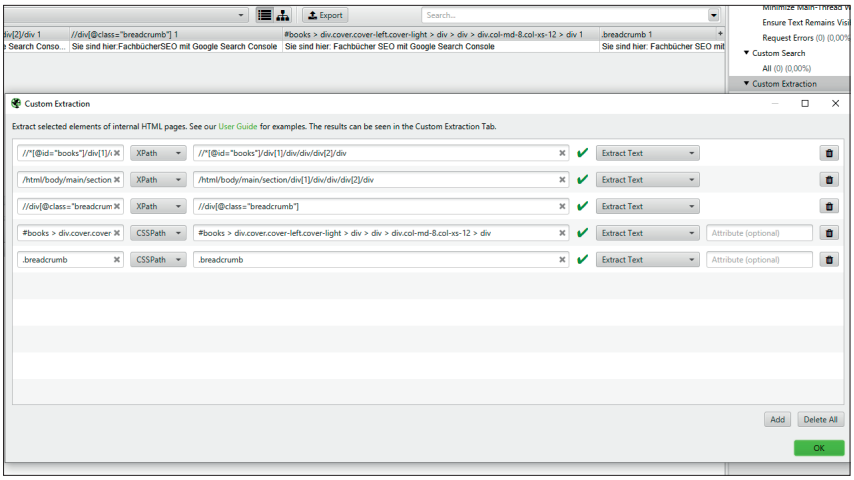

Abb. 8: Im Screaming Frog können Sie unter Configuration => Custom => Extraction definieren, was Sie wie aus der Website extrahieren möchten

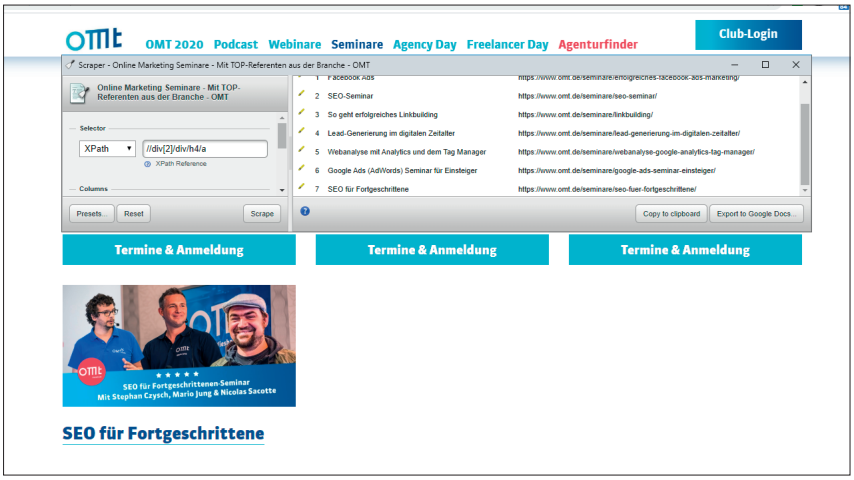

Abb. 9: Durch die Integration ins Rechtsklick-Menü des Chrome-Browsers lassen sich mit der Scrape-Similar-Erweiterung schnell alle anstehenden Seminare der Seite extrahieren

ist dies die richtige Auswahl.

Dazu gibt es die Möglichkeit, den sogenannten Function Value zu extrahieren. Damit können Sie über Angaben wie count(//h1) zählen, wie viele H1-Überschriften auf einer Seite enthalten sind. Übrigens: Unter *www.screamingfrog.co.uk/web-scraping/* finden Sie einige weitere Beispiel-XPaths.

Datenscraping mit Google Sheets

Das beliebte Online-Tabellenkalkulationsprogramm bietet über die Funktion =**IMPORTXML(„Adresse“;“XPath“)** die Möglichkeit, Daten von URLs auszulesen. So würden über die Formel =IMPORTXML(„https://www.websiteboosting.com/ „; „//a/@href“) alle Links extrahiert werden, die auf der Startseite zu finden sind.

Importxml macht Google Sheets zu einem sehr mächtigen Tool und es ist u. a. möglich, damit Websites zu überwachen. Wenn Sie beispielsweise kontrollieren möchten, ob eine Seite durch die Noindex-Robotsangabe von der Indexierung ausgeschlossen ist, dann können Sie dies mit der entsprechenden Konfiguration über Google Sheets lösen. Unter *www.deptagency.com/de-de/download/onpage-monitoring-tool/* finden Sie eine kostenlose Monitoring-Lösung für Websites basierend auf Google Sheets.

Scraping-Tools direkt im Browser

Sie möchten nicht zwischen Ihrem Browser und weiteren Programmen hin- und herwechseln, um Daten zu extrahieren? Dann nutzen Sie einfach

TIPP

Über den Screaming Frog finden Sie zwei Test- und Anwendungsberichte online zum Nachlesen aus den Ausgaben 52 und 53 unter *http://einfach.st/frog1* und *http://einfach.st/frog2*.

eine Erweiterung wie das Chrome-Plugin **Scrape Similar**. Nach Angabe des gewünschten XPaths sind die Daten in der Vorschau zu sehen und können umgehend weiterverarbeitet werden.

Als Alternativen können Online-Tools wie import.io (*www.import.io/*) oder parsehub.com (*www.parsehub.com/*) verwendet werden. Diese können zeitgesteuert Daten (immer wieder neu) extrahieren.

Erleichtern auch Sie sich die Kopierarbeit dank XPath und CSSPath

Sie haben nun das benötigte Wissen zur Hand, um zukünftig schneller an Daten zu gelangen. Richtig eingesetzt hilft Ihnen Scraping dabei, Ihre Website umfassend(er) im Blick zu haben und Veränderungen zu überwachen.

Crawler wie der Screaming Frog lesen viele interessante Daten bereits ohne weitere Konfiguration aus – alles, was Sie darüber hinaus benötigen, bekommen Sie mit den entsprechenden XPath- oder CSSPath-Angaben. Nun dann: Setzen Sie Ihr Scraping-Wissen direkt in der Praxis ein!