Kostenloses eBook

LERNEN

Jsoup

Free unaffiliated eBook created from
Stack Overflow contributors.

#jsoup

Inhaltsverzeichnis

Über

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: jsoup

It is an unofficial and free Jsoup ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official Jsoup.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Kapitel 1: Erste Schritte mit Jsoup

Bemerkungen

Jsoup ist eine HTML-Analyse- und Datenextraktionsbibliothek für Java, die auf Flexibilität und Benutzerfreundlichkeit ausgerichtet ist. Es kann verwendet werden, um spezifische Daten aus HTML-Seiten zu extrahieren, was gemeinhin als "Web-Scraping" bezeichnet wird, sowie zum Ändern des Inhalts von HTML-Seiten und zum "sauberen" nicht vertrauenswürdigen HTML-Code mit einer Whitelist der zulässigen Tags und Attribute.

JavaScript-Unterstützung

Jsoup unterstützt kein JavaScript. Daher können dynamisch generierte Inhalte oder Inhalte, die nach dem Laden der Seite zur Seite hinzugefügt werden, nicht aus der Seite extrahiert werden. Wenn Sie Inhalte extrahieren möchten, die der Seite mit JavaScript hinzugefügt werden, gibt *es* einige alternative Optionen:

- Verwenden Sie eine Bibliothek, die JavaScript unterstützt, wie z. B. Selenium, das zum Laden von Seiten einen Webbrowser verwendet, oder HtmlUnit.
- Reverse Engineering, wie die Seite ihre Daten lädt. Normalerweise machen Webseiten, die Daten dynamisch laden, dies über AJAX. Daher können Sie auf der Registerkarte "Netzwerk" der Entwicklerwerkzeuge Ihres Browsers nachsehen, woher die Daten geladen werden, und diese URLs dann in Ihrem eigenen Code verwenden. Weitere Informationen finden Sie unter Kratzen von AJAX-Seiten .

Offizielle Website und Dokumentation

Bei **Jsoup.org** finden Sie verschiedene Ressourcen zu **Jsoup** , einschließlich Javadoc , Verwendungsbeispiele im Jsoup-Kochbuch und JAR-Downloads . **Im GitHub-Repository finden Sie** Quellcode, Probleme und Pull-Anforderungen.

Herunterladen

Jsoup ist auf Maven als `org.jsoup.jsoup:jsoup` , Wenn Sie Gradle verwenden (z. B. mit Android Studio), können Sie es zu Ihrem Projekt hinzufügen, indem Sie Folgendes zum Abschnitt `build.gradle` Abhängigkeiten hinzufügen:

```
compile 'org.jsoup:jsoup:1.8.3'
```

Wenn Sie Ant (Eclipse) verwenden, fügen Sie dem Abschnitt mit den POM-Abhängigkeiten Folgendes hinzu:

```
<dependency>
  <!-- jsoup HTML parser library @ http://jsoup.org/ -->
  <groupId>org.jsoup</groupId>
  <artifactId>jsoup</artifactId>
  <version>1.8.3</version>
</dependency>
```

Jsoup ist auch als **herunterladbare JAR** für andere Umgebungen verfügbar.

Versionen

Ausführung	Veröffentlichungsdatum
1.9.2	2016-05-17
1.8.3	2015-08-02

Examples

Extrahieren Sie die URLs und Titel von Links

Mit Jsoup können Sie auf einfache Weise alle Links von einer Webseite extrahieren. In diesem Fall können Sie Jsoup verwenden, um nur bestimmte Links zu extrahieren, die hier in einem `h3` Header auf einer Seite angezeigt werden sollen. Wir können auch den Text der Links erhalten.

```
Document doc = Jsoup.connect("http://stackoverflow.com").userAgent("Mozilla").get();
for (Element e: doc.select("a.question-hyperlink")) {
    System.out.println(e.attr("abs:href"));
    System.out.println(e.text());
    System.out.println();
}
```

Dies ergibt die folgende Ausgabe:

```
http://stackoverflow.com/questions/12920296/past-5-week-calculation-in-webi-bo-4-0
Past 5 week calculation in WEBI (BO 4.0)?

http://stackoverflow.com/questions/36303701/how-to-get-information-about-the-visualized-
elements-in-listview
How to get information about the visualized elements in listview?

[...]
```

Was passiert hier:

- Zuerst erhalten wir das HTML-Dokument von der angegebenen URL. Dieser Code setzt auch den User Agent-Header der Anforderung auf "Mozilla", sodass die Website die Seite bereitstellt, die sie normalerweise für Browser bereitstellt.

- Verwenden Sie dann `select(...)` und eine for-Schleife, um alle Links zu Stack Overflow-Fragen zu erhalten, in diesem Fall Links, die den Klassenfrage `question-hyperlink` .
- Drucken Sie den Text jedes Links mit `.text()` und den Link des Links mit `attr("abs:href")` . In diesem Fall verwenden wir `abs:` um die *absolute* URL zu erhalten, dh. mit der Domäne und dem Protokoll enthalten.

Vollständige URL aus teilweise HTML extrahieren

Wenn Sie nur den Attributwert eines Links auswählen: href gibt die relative URL zurück.

```
   String bodyFragment =
          "<div><a href=\"/documentation\">Stack Overflow Documentation</a></div>";


    Document doc = Jsoup.parseBodyFragment(bodyFragment);
    String link = doc
            .select("div > a")
            .first()
            .attr("href");

    System.out.println(link);
```

Ausgabe

```
/documentation
```

Durch das Bestehen des Basis - URI in die `parse` - Methode und unter Verwendung der `absUrl` Methode anstelle von `attr` , können wir die vollständige URL extrahieren.

```
    Document doc = Jsoup.parseBodyFragment(bodyFragment, "http://stackoverflow.com");

    String link = doc
                .select("div > a")
                .first()
                .absUrl("href");

    System.out.println(link);
```

Ausgabe

```
http://stackoverflow.com/documentation
```

Extrahieren Sie die Daten aus der HTML-Dokumentdatei

Mit Jsoup können Sie Daten aus einer lokalen **Datei** , die HTML enthält, bearbeiten oder extrahieren. `filePath` ist der Pfad einer Datei auf der Festplatte. `ENCODING` ist der gewünschte Zeichensatzname, z. B. "Windows-31J". Es ist optional.

```
 // load file
```

```
    File inputFile = new File(filePath);
    // parse file as HTML document
    Document doc = Jsoup.parse(filePath, ENCODING);
    // select element by <a>
    Elements elements = doc.select("a");
```

Erste Schritte mit Jsoup online lesen: https://riptutorial.com/de/jsoup/topic/297/erste-schritte-mit-jsoup

Kapitel 2: HTML-Ausgabe formatieren

Parameter

Parameter	Detail
`boolean outline()`	Abrufen, wenn der Gliederungsmodus aktiviert ist. Standardeinstellung ist "false". Wenn aktiviert, betrachten die HTML-Ausgabemethoden alle Tags als Block.
`Document.OutputSettings outline(boolean)`	Aktivieren oder deaktivieren Sie den HTML-Gliederungsmodus.

Bemerkungen

Jsoup 1.9.2 API

Examples

Alle Elemente als Block anzeigen

Standardmäßig zeigt Jsoup nur Elemente auf Blockebene mit einem Zeilenumbruch an. Inline-Elemente werden ohne Zeilenumbruch angezeigt.

Bei einem Body-Fragment mit Inline-Elementen:

```
<select name="menu">
    <option value="foo">foo</option>
    <option value="bar">bar</option>
</select>
```

Drucken mit Jsoup:

```
Document doc = Jsoup.parse(html);

System.out.println(doc.html());
```

Ergebnisse in:

```
<html>
 <head></head>
 <body>
  <select name="menu"> <option value="foo">foo</option> <option value="bar">bar</option>
</select>
 </body>
</html>
```

Um die Ausgabe mit jedem als Blockelement behandelten Element anzuzeigen, muss die `outline` in den `OutputSettings` des Dokuments `OutputSettings` .

```
Document doc = Jsoup.parse(html);

doc.outputSettings().outline(true);

System.out.println(doc.html());
```

Ausgabe

```
<html>
 <head></head>
 <body>
  <select name="menu">
   <option value="foo">foo</option>
   <option value="bar">bar</option>
  </select>
 </body>
</html>
```

Source: JSoup - Formatieren der <Option> -Elemente

HTML-Ausgabe formatieren online lesen: https://riptutorial.com/de/jsoup/topic/5954/html-ausgabe-formatieren

Kapitel 3: Javascript generierte Seiten analysieren

Examples

Analysieren von JavaScript generierte Seite mit Jsoup und HtmUnit

page.html - Quellcode

```
<html>
<head>
    <script src="loadData.js"></script>
</head>
<body onLoad="loadData()">
    <div class="container">
        <table id="data" border="1">
            <tr>
                <th>col1</th>
                <th>col2</th>
            </tr>
        </table>
    </div>
</body>
</html>
```

loadData.js

```
    // append rows and cols to table.data in page.html
    function loadData() {
        data = document.getElementById("data");
        for (var row = 0; row < 2; row++) {
            var tr = document.createElement("tr");
            for (var col = 0; col < 2; col++) {
                td = document.createElement("td");
                td.appendChild(document.createTextNode(row + "." + col));
                tr.appendChild(td);
            }
            data.appendChild(tr);
        }
    }
```

page.html beim Laden in den Browser

Col1	Col2
0,0	0,1
1,0	1.1

Verwenden Sie jsoup, um page.html nach Col-Daten zu analysieren

```
    // load source from file
    Document doc = Jsoup.parse(new File("page.html"), "UTF-8");

    // iterate over row and col
    for (Element row : doc.select("table#data > tbody > tr"))

        for (Element col : row.select("td"))

            // print results
            System.out.println(col.ownText());
```

Ausgabe

(leeren)

Was ist passiert?

Jsoup analysiert den vom Server gelieferten (oder in diesem Fall aus der Datei geladenen) Quellcode. Es werden keine clientseitigen Aktionen wie die Manipulation von JavaScript oder CSS-DOM aufgerufen. In diesem Beispiel werden die Zeilen und Spalten niemals an die Datentabelle angehängt.

Wie kann ich meine Seite im Browser rendern?

```
    // load page using HTML Unit and fire scripts
    WebClient webClient = new WebClient();
    HtmlPage myPage = webClient.getPage(new File("page.html").toURI().toURL());

    // convert page to generated HTML and convert to document
    doc = Jsoup.parse(myPage.asXml());

    // iterate row and col
    for (Element row : doc.select("table#data > tbody > tr"))

        for (Element col : row.select("td"))

            // print results
            System.out.println(col.ownText());

    // clean up resources
    webClient.close();
```

Ausgabe

```
0.0
0.1
1.0
1.1
```

Javascript generierte Seiten analysieren online lesen:
https://riptutorial.com/de/jsoup/topic/4632/javascript-generierte-seiten-analysieren

Kapitel 4: Mit Jsoup in Websites einloggen

Examples

Eine einfache Authentifizierungs-POST-Anforderung mit Jsoup

Nachfolgend wird eine einfache POST-Anforderung mit Authentifizierungsdaten beschrieben. Beachten Sie, dass das Feld `username` und `password` je nach Website unterschiedlich ist:

```
final String USER_AGENT = "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/51.0.2704.103 Safari/537.36";
Connection.Response loginResponse = Jsoup.connect("yourWebsite.com/loginUrl")
                                         .userAgent(USER_AGENT)
                                         .data("username", "yourUsername")
                                         .data("password", "yourPassword")
                                         .method(Method.POST)
                                         .execute();
```

Eine umfassendere Authentifizierungs-POST-Anforderung mit Jsoup

Die meisten Websites erfordern ein viel komplizierteres Verfahren als das oben gezeigte.

Übliche Schritte zum Einloggen in eine Website sind:

1. Holen Sie sich das einzigartige `cookie` aus dem Anmeldeformular.
2. Überprüfen Sie das Anmeldeformular, um die Ziel-URL für die Authentifizierungsanforderung zu ermitteln
3. Durchsuchen Sie das Anmeldeformular, um zu überprüfen, ob ein `security token` mit Benutzername und Kennwort gesendet werden muss.
4. Senden Sie die Anfrage.

Nachfolgend finden Sie eine Beispielanfrage, mit der Sie auf der GitHub- Website angemeldet werden

```
// # Constants used in this example
final String USER_AGENT = "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/51.0.2704.103 Safari/537.36";
final String LOGIN_FORM_URL = "https://github.com/login";
final String LOGIN_ACTION_URL = "https://github.com/session";
final String USERNAME = "yourUsername";
final String PASSWORD = "yourPassword";

// # Go to login page and grab cookies sent by server
Connection.Response loginForm = Jsoup.connect(LOGIN_FORM_URL)
                                     .method(Connection.Method.GET)
                                     .userAgent(USER_AGENT)
                                     .execute();
Document loginDoc = loginForm.parse(); // this is the document containing response html
HashMap<String, String> cookies = new HashMap<>(loginForm.cookies()); // save the cookies to
be passed on to next request
```

```
// # Prepare login credentials
String authToken = loginDoc.select("#login > form > div:nth-child(1) >
input[type=\"hidden\"]:nth-child(2)")
                           .first()
                           .attr("value");

HashMap<String, String> formData = new HashMap<>();
formData.put("commit", "Sign in");
formData.put("utf8", "e2 9c 93");
formData.put("login", USERNAME);
formData.put("password", PASSWORD);
formData.put("authenticity_token", authToken);

// # Now send the form for login
Connection.Response homePage = Jsoup.connect(LOGIN_ACTION_URL)
     .cookies(cookies)
     .data(formData)
     .method(Connection.Method.POST)
     .userAgent(USER_AGENT)
     .execute();

System.out.println(homePage.parse().html());
```

Protokollierung mit FormElement

In diesem Beispiel melden wir uns mit der FormElement- Klasse bei der GitHub- Website an.

```
// # Constants used in this example
final String USER_AGENT = "Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/51.0.2704.103 Safari/537.36";
final String LOGIN_FORM_URL = "https://github.com/login";
final String USERNAME = "yourUsername";
final String PASSWORD = "yourPassword";

// # Go to login page
Connection.Response loginFormResponse = Jsoup.connect(LOGIN_FORM_URL)
                                             .method(Connection.Method.GET)
                                             .userAgent(USER_AGENT)
                                             .execute();

// # Fill the login form
// ## Find the form first...
FormElement loginForm = (FormElement)loginFormResponse.parse()
                                        .select("div#login > form").first();
checkElement("Login Form", loginForm);

// ## ... then "type" the username ...
Element loginField = loginForm.select("#login_field").first();
checkElement("Login Field", loginField);
loginField.val(USERNAME);

// ## ... and "type" the password
Element passwordField = loginForm.select("#password").first();
checkElement("Password Field", passwordField);
passwordField.val(PASSWORD);


// # Now send the form for login
Connection.Response loginActionResponse = loginForm.submit()
```

```
            .cookies(loginFormResponse.cookies())
            .userAgent(USER_AGENT)
            .execute();

System.out.println(loginActionResponse.parse().html());

public static void checkElement(String name, Element elem) {
    if (elem == null) {
        throw new RuntimeException("Unable to find " + name);
    }
}
```

Alle Formulardaten werden für uns von der FormElement-Klasse verarbeitet (sogar die Erkennung der Formularmethode). Eine fertige Verbindung wird beim Aufrufen der FormElement # Submit-Methode erstellt. Alles, was wir tun müssen, ist, diese Verbindung mit zusätzlichen Kopfzeilen (Cookies, User-Agent usw.) herzustellen und auszuführen.

Mit Jsoup in Websites einloggen online lesen: https://riptutorial.com/de/jsoup/topic/4631/mit-jsoup-in-websites-einloggen

Kapitel 5: Selektoren

Bemerkungen

Ein Selektor ist eine Kette einfacher Selektoren, die durch Kombinatoren getrennt sind. Selektoren unterscheiden nicht zwischen Groß- und Kleinschreibung (einschließlich gegen Elemente, Attribute und Attributwerte).

Der Universalselektor (*) ist implizit, wenn kein Elementselektor bereitgestellt wird (dh * .header und .header sind gleichwertig).

Muster	Streichhölzer	Beispiel
`*`	irgendein Element	`*`
`tag`	Elemente mit dem angegebenen Tagnamen	`div`
`ns|E`	Elemente vom Typ E im Namensraum ns	`fb|name finds <fb:name> elements`
`#id`	Elemente mit Attribut-ID "ID"	`div#wrap, #logo`
`.class`	Elemente mit einem Klassennamen "Klasse"	`div.left, .result`
`[attr]`	Elemente mit einem Attribut "attr" (mit einem beliebigen Wert)	`a[href], [title]`
`[^attrPrefix]`	Elemente mit einem Attributnamen, der mit "attrPrefix" beginnt. Verwenden Sie diese Option, um Elemente mit HTML5-Datasets zu suchen	`[^data-], div[^data-]`
`[attr=val]`	Elemente mit einem Attribut "attr" und einem Wert gleich "val"	`img[width=500], a[rel=nofollow]`

Muster	Streichhölzer	Beispiel
`[attr="val"]`	Elemente mit einem Attribut "attr" und einem Wert gleich "val"	`span[hello="Cleveland"][goodbye="Columbus"], a[rel="nofollow"]`
`[attr^=valPrefix]`	Elemente mit einem Attribut "attr" und einem Wert, der mit "valPrefix" beginnt	`a[href^=http:]`
`[attr$=valSuffix]`	Elemente mit einem Attribut "attr" und einem Wert, der mit "valSuffix" endet	`img[src$=.png]`
`[attr*=valContaining]`	Elemente mit einem Attribut "attr" und einem Wert, der "valContaining" enthält	`a[href*=/search/]`
`[attr~=regex]`	Elemente mit einem Attribut "attr" und einem Wert, der dem regulären Ausdruck entspricht	`img[src~=(?i)\.(png|jpe?g)]`
	Das Obige kann in beliebiger Reihenfolge kombiniert werden	`div.header[title]`

Selektor vollständige Referenz

Examples

Elemente mit CSS-Selektoren auswählen

```
String html = "<!DOCTYPE html>" +
              "<html>" +
                "<head>" +
                  "<title>Hello world!</title>" +
                "</head>" +
                "<body>" +
                  "<h1>Hello there!</h1>" +
                  "<p>First paragraph</p>" +
                  "<p class=\"not-first\">Second paragraph</p>" +
```

```
                  "<p class=\"not-first third\">Third <a href=\"page.html\">paragraph</a></p>"
+
                "</body>" +
              "</html>";

// Parse the document
Document doc = Jsoup.parse(html);

// Get document title
String title = doc.select("head > title").first().text();
System.out.println(title); // Hello world!

Element firstParagraph = doc.select("p").first();

// Get all paragraphs except from the first
Elements otherParagraphs = doc.select("p.not-first");
// Same as
otherParagraphs = doc.select("p");
otherParagraphs.remove(0);

// Get the third paragraph (second in the list otherParagraphs which
// excludes the first paragraph)
Element thirdParagraph = otherParagraphs.get(1);
// Alternative:
thirdParagraph = doc.select("p.third");

// You can also select within elements, e.g. anchors with a href attribute
// within the third paragraph.
Element link = thirdParagraph.select("a[href]");
// or the first <h1> element in the document body
Element headline = doc.select("body").first().select("h1").first();
```

Eine detaillierte Übersicht der unterstützten Selektoren finden Sie hier .

Twitter Markup extrahieren

```
    // Twitter markup documentation:
    // https://dev.twitter.com/cards/markup
    String[] twitterTags = {
            "twitter:site",
            "twitter:site:id",
            "twitter:creator",
            "twitter:creator:id",
            "twitter:description",
            "twitter:title",
            "twitter:image",
            "twitter:image:alt",
            "twitter:player",
            "twitter:player:width",
            "twitter:player:height",
            "twitter:player:stream",
            "twitter:app:name:iphone",
            "twitter:app:id:iphone",
            "twitter:app:url:iphone",
            "twitter:app:name:ipad",
            "twitter:app:id:ipad",
            "twitter:app:url:ipadt",
            "twitter:app:name:googleplay",
            "twitter:app:id:googleplay",
```

```
            "twitter:app:url:googleplay"
    };

    // Connect to URL and extract source code
    Document doc = Jsoup.connect("http://stackoverflow.com/").get();

    for (String twitterTag : twitterTags) {

        // find a matching meta tag
        Element meta = doc.select("meta[name=" + twitterTag + "]").first();

        // if found, get the value of the content attribute
        String content = meta != null ? meta.attr("content") : "";

        // display results
        System.out.printf("%s = %s%n", twitterTag, content);
    }
```

Ausgabe

```
twitter:site =
twitter:site:id =
twitter:creator =
twitter:creator:id =
twitter:description = Q&A for professional and enthusiast programmers
twitter:title = Stack Overflow
twitter:image =
twitter:image:alt =
twitter:player =
twitter:player:width =
twitter:player:height =
twitter:player:stream =
twitter:app:name:iphone =
twitter:app:id:iphone =
twitter:app:url:iphone =
twitter:app:name:ipad =
twitter:app:id:ipad =
twitter:app:url:ipadt =
twitter:app:name:googleplay =
twitter:app:id:googleplay =
twitter:app:url:googleplay =
```

Selektoren online lesen: https://riptutorial.com/de/jsoup/topic/515/selektoren

Kapitel 6: Web-Crawling mit Jsoup

Examples

Extrahieren von E-Mail-Adressen und Links zu anderen Seiten

Jsoup kann zum Extrahieren von Links und E-Mail-Adressen von einer Webseite verwendet werden, daher "Web-E-Mail-Adressensammler-Bot". Zunächst verwendet dieser Code einen regulären Ausdruck, um die E-Mail-Adressen zu extrahieren. Anschließend werden mithilfe von von Jsoup bereitgestellten Methoden URLs von Links extrahiert Die Seite.

```
public class JSoupTest {

    public static void main(String[] args) throws IOException {
        Document doc =
Jsoup.connect("http://stackoverflow.com/questions/15893655/").userAgent("Mozilla").get();

        Pattern p = Pattern.compile("[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\\.[a-zA-Z0-9-.]+");
        Matcher matcher = p.matcher(doc.text());
        Set<String> emails = new HashSet<String>();
        while (matcher.find()) {
            emails.add(matcher.group());
        }

        Set<String> links = new HashSet<String>();

        Elements elements = doc.select("a[href]");
        for (Element e : elements) {
            links.add(e.attr("href"));
        }

        System.out.println(emails);
        System.out.println(links);

    }

}
```

Dieser Code kann auch leicht erweitert werden, um auch diese URLs rekursiv aufzurufen und Daten aus verknüpften Seiten zu extrahieren. Es könnte auch leicht mit einem anderen Regex verwendet werden, um andere Daten zu extrahieren.

(Bitte kein Spammer werden!)

Extrahieren von JavaScript-Daten mit Jsoup

In diesem Beispiel werden wir versuchen, JavaScript-Daten zu finden, die `backgroundColor:'#FFF'` enthalten `backgroundColor:'#FFF'` . Dann ändern wir den Wert von backgroundColor `'#FFF' '#ddd'` . Dieser Code verwendet die `getWholeData()` und `setWholeData()` , um JavaScript-Daten zu `setWholeData()` . Alternativ kann die `html()` -Methode verwendet werden, um JavaScript-Daten abzurufen.

```
// create HTML with JavaScript data
    StringBuilder html = new StringBuilder();
    html.append("<!DOCTYPE html> <html> <head> <title>Hello Jsoup!</title>");
    html.append("<script>");
    html.append("StackExchange.docs.comments.init({");
    html.append("highlightColor: '#F4A83D',");
    html.append("backgroundColor:'#FFF',");
    html.append("});");
    html.append("</script>");
    html.append("<script>");
    html.append("document.write(<style type='text/css'>div,iframe { top: 0; position:absolute;
}</style>');");
    html.append("</script>\n");
    html.append("</head><body></body> </html>");

    // parse as HTML document
    Document doc = Jsoup.parse(html.toString());

    String defaultBackground = "backgroundColor:'#FFF'";
    // get <script>
    for (Element scripts : doc.getElementsByTag("script")) {
        // get data from <script>
        for (DataNode dataNode : scripts.dataNodes()) {
            // find data which contains backgroundColor:'#FFF'
            if (dataNode.getWholeData().contains(defaultBackground)) {
                // replace '#FFF' -> '#ddd'
                String newData = dataNode.getWholeData().replaceAll(defaultBackground,
"backgroundColor:'#ddd'");
                // set new data contents
                dataNode.setWholeData(newData);
            }
        }
    }
    System.out.println(doc.toString());
```

Ausgabe

```
<script>StackExchange.docs.comments.init({highlightColor:
'#F4A83D',backgroundColor:'#ddd',});</script>
```

Extrahieren aller URLs von einer Website mithilfe von JSoup (Rekursion)

In diesem Beispiel extrahieren wir alle Weblinks von einer Website. Ich verwende `http://stackoverflow.com/` zur Illustration. Hier wird Rekursion verwendet, bei der die Seite jedes erhaltenen Links auf Vorhandensein eines `anchor tag` analysiert wird und dieser Link erneut der gleichen Funktion übergeben wird.

Die Bedingung `if(add && this_url.contains(my_site))` begrenzt die Ergebnisse nur **auf Ihre Domain** .

```
    import java.io.IOException;
    import java.util.HashSet;
    import java.util.Set;
    import org.jsoup.Jsoup;
    import org.jsoup.nodes.Document;
    import org.jsoup.select.Elements;
```

```
    public class readAllLinks {

        public static Set<String> uniqueURL = new HashSet<String>();
        public static String my_site;

        public static void main(String[] args) {

            readAllLinks obj = new readAllLinks();
            my_site = "stackoverflow.com";
            obj.get_links("http://stackoverflow.com/");
        }

        private void get_links(String url) {
            try {
                Document doc = Jsoup.connect(url).userAgent("Mozilla").get();
                Elements links = doc.select("a");

                if (links.isEmpty()) {
                   return;
                }

                links.stream().map((link) -> link.attr("abs:href")).forEachOrdered((this_url)
-> {
                    boolean add = uniqueURL.add(this_url);
                    if (add && this_url.contains(my_site)) {
                        System.out.println(this_url);
                        get_links(this_url);
                    }
                });

            } catch (IOException ex) {

            }

        }
    }
```

Die Ausführung des Programms kann je nach Website viel Zeit in Anspruch nehmen. Der obige Code kann erweitert werden, um Daten (wie Seitentitel, Text oder Bilder) von einer bestimmten Website zu extrahieren. Ich würde Ihnen empfehlen, die Nutzungsbedingungen des Unternehmens durchzulesen, bevor Sie die Website verschrotten.

In diesem Beispiel wird die JSoup-Bibliothek verwendet, um die Links `your_url/sitemap.xml` . Sie können die Links auch mithilfe `your_url/sitemap.xml` .

Web-Crawling mit Jsoup online lesen: https://riptutorial.com/de/jsoup/topic/319/web-crawling-mit-jsoup

Credits

S. No	Kapitel	Contributors
1	Erste Schritte mit Jsoup	Alice, Community, Jeffrey Bosboom, JonasCz, Zack Teater
2	HTML-Ausgabe formatieren	Zack Teater
3	Javascript generierte Seiten analysieren	Zack Teater
4	Mit Jsoup in Websites einloggen	Joel Min, JonasCz, Stephan
5	Selektoren	JonasCz, Stephan, still_learning, Zack Teater
6	Web-Crawling mit Jsoup	Alice, JonasCz, r_D, RamenChef