EBook Gratuito

APPRENDIMENTO

Jsoup

Free unaffiliated eBook created from
Stack Overflow contributors.

#jsoup

Sommario

Di

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

Capitolo 1: Iniziare con Jsoup

Osservazioni

Jsoup è una libreria di analisi e estrazione di dati HTML per Java, incentrata sulla flessibilità e sulla facilità d'uso. Può essere usato per estrarre dati specifici da pagine HTML, che è comunemente noto come "scraping web", così come modificare il contenuto di pagine HTML e "ripulire" HTML non fidato con una lista bianca di tag e attributi consentiti.

Supporto JavaScript

Jsoup non supporta JavaScript e, a causa di ciò, qualsiasi contenuto o contenuto generato dinamicamente che viene aggiunto alla pagina dopo il caricamento della pagina non può essere estratto dalla pagina. Se è necessario estrarre il contenuto che viene aggiunto alla pagina con JavaScript, ci *sono* alcune opzioni alternative:

- Utilizzare una libreria che supporti JavaScript, come Selenium, che utilizza un browser Web effettivo per caricare pagine o HtmlUnit.
- Reverse engineer come la pagina carica i suoi dati. In genere, le pagine Web che caricano i dati in modo dinamico lo fanno tramite AJAX e, quindi, è possibile consultare la scheda di rete degli strumenti di sviluppo del browser per vedere dove vengono caricati i dati e quindi utilizzare tali URL nel proprio codice. Guarda come raschiare le pagine AJAX per maggiori dettagli.

Sito web ufficiale e documentazione

Puoi trovare varie risorse relative a **Jsoup** su **jsoup.org** , inclusi Javadoc , esempi di utilizzo nel libro di ricette Jsoup e download JAR . Vedi **il repository GitHub** per il codice sorgente, i problemi e le richieste di pull.

Scaricare

Jsoup è disponibile su Maven come `org.jsoup.jsoup:jsoup` , Se stai utilizzando Gradle (ad esempio con Android Studio), puoi aggiungerlo al tuo progetto aggiungendo quanto segue alla tua sezione delle dipendenze `build.gradle` :

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

Se stai usando Ant (Eclipse), aggiungi quanto segue alla sezione delle dipendenze di POM:

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

```
  <artifactId>jsoup</artifactId>
  <version>1.8.3</version>
</dependency>
```

Jsoup è anche disponibile come **JAR scaricabile** per altri ambienti.

Versioni

Versione	Data di rilascio
1.9.2	2016/05/17
1.8.3	2015/08/02

Examples

Estrai gli URL e i titoli dei link

Jsoup può essere usato per estrarre facilmente tutti i link da una pagina web. In questo caso, possiamo usare Jsoup per estrarre solo link specifici che vogliamo, qui, quelli in un'intestazione `h3` su una pagina. Possiamo anche ottenere il testo dei collegamenti.

```
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();
}
```

Questo dà il seguente risultato:

```
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?

[...]
```

Cosa sta succedendo qui:

- Innanzitutto, otteniamo il documento HTML dall'URL specificato. Questo codice imposta anche l'intestazione User Agent della richiesta su "Mozilla", in modo che il sito web serva la pagina che di solito servirà ai browser.
- Quindi, utilizzare `select(...)` e un ciclo for per ottenere tutti i collegamenti alle domande Stack Overflow, in questo caso i collegamenti che presentano il `question-hyperlink` della

classe.

- Stampa il testo di ogni link con `.text()` e il href del link con `attr("abs:href")` . In questo caso, usiamo `abs:` per ottenere l'URL *assoluto* , es. con il dominio e il protocollo inclusi.

Estrai URL completo da HTML parziale

Selezionando solo il valore dell'attributo di un link: href restituirà l'URL relativo.

```
   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);
```

Produzione

```
/documentation
```

Passando l'URI di base nel metodo di `parse` e utilizzando il metodo `absUrl` anziché `attr` , possiamo estrarre l'URL completo.

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

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

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

Produzione

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

Estrai i dati dal file di documento HTML

Jsoup può essere utilizzato per manipolare o estrarre dati da un **file** su local che contiene HTML. `filePath` è il percorso di un file su disco. `ENCODING` è il nome del set di caratteri desiderato, ad esempio "Windows-31J". È opzionale

```
 // 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");
```

Leggi Iniziare con Jsoup online: https://riptutorial.com/it/jsoup/topic/297/iniziare-con-jsoup

Capitolo 2: Accesso a siti Web con Jsoup

Examples

Una semplice richiesta POST di autenticazione con Jsoup

Di seguito è riportata una semplice richiesta POST con i dati di autenticazione, notare che il campo `username` e `password` varieranno in base al sito Web:

```
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();
```

Una richiesta POST di autenticazione più completa con Jsoup

La maggior parte dei siti Web richiede un processo molto più complicato di quello illustrato sopra.

I passaggi comuni per accedere a un sito Web sono:

1. Ottieni il `cookie` unico dal modulo di accesso iniziale.
2. Ispezionare il modulo di accesso per vedere quale sia l'url di destinazione per la richiesta di autenticazione
3. Analizzare il modulo di accesso per verificare eventuali `security token` che devono essere inviati insieme a nome utente e password.
4. Invia la richiesta

Di seguito è riportata una richiesta di esempio che ti consente di accedere al sito Web GitHub

```
// # 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());
```

Registrazione con FormElement

In questo esempio, effettueremo l'accesso al sito Web GitHub utilizzando la classe FormElement .

```
// # 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);
    }
}
```

Tutti i dati del modulo sono gestiti dalla classe FormElement per noi (anche il metodo di rilevamento del modulo). Una connessione già pronta viene creata quando si richiama il metodo di invio # FormElement . Tutto ciò che dobbiamo fare è completare questa connessione con intestazioni aggiuntive (cookie, user-agent, ecc.) Ed eseguirla.

Leggi Accesso a siti Web con Jsoup online: https://riptutorial.com/it/jsoup/topic/4631/accesso-a-siti-web-con-jsoup

Capitolo 3: Analisi delle pagine generate da Javascript

Examples

Analisi della pagina generata da JavaScript con Jsoup e HtmUnit

page.html - codice sorgente

```
<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 quando caricato nel browser

col1	col2
0.0	0.1
1.0	1.1

Usando jsoup per analizzare page.html per i dati col

```
    // 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());
```

Produzione

(vuoto)

Quello che è successo?

Jsoup analizza il codice sorgente come consegnato dal server (o in questo caso caricato dal file). Non invoca azioni lato client come la manipolazione del DOM JavaScript o CSS. In questo esempio, le righe e le colonne non vengono mai aggiunte alla tabella dati.

Come analizzare la mia pagina come renderizzata nel browser?

```
    // 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();
```

Produzione

```
0.0
0.1
1.0
1.1
```

Leggi Analisi delle pagine generate da Javascript online:
https://riptutorial.com/it/jsoup/topic/4632/analisi-delle-pagine-generate-da-javascript

Capitolo 4: Formattazione dell'output HTML

Parametri

Parametro	Dettaglio
`boolean outline()`	Ottieni se la modalità struttura è abilitata. Il valore predefinito è falso. Se abilitato, i metodi di output HTML considereranno tutti i tag come blocchi.
`Document.OutputSettings outline(boolean)`	Abilita o disabilita la modalità struttura HTML.

Osservazioni

API Jsoup 1.9.2

Examples

Mostra tutti gli elementi come blocchi

Per impostazione predefinita, Jsoup visualizza solo gli elementi a livello di blocco con un'interruzione di riga finale. Gli elementi in linea vengono visualizzati senza interruzione di riga.

Dato un frammento del corpo, con elementi in linea:

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

Stampa con Jsoup:

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

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

Risultati in:

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

Per visualizzare l'output con ogni elemento trattato come elemento di blocco, l'opzione di `outline` deve essere abilitata nelle `OutputSettings` di `OutputSettings` del documento.

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

doc.outputSettings().outline(true);

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

Produzione

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

Fonte: JSoup - Formattazione degli elementi <option>

Leggi Formattazione dell'output HTML online:
https://riptutorial.com/it/jsoup/topic/5954/formattazione-dell-output-html

Capitolo 5: I selettori

Osservazioni

Un selettore è una catena di selettori semplici, separati da combinatori. I selettori non fanno distinzione tra maiuscole e minuscole (inclusi elementi, attributi e valori degli attributi).

Il selettore universale (*) è implicito quando non viene fornito alcun selettore di elementi (ad esempio * .header e .header è equivalente).

Modello	fiammiferi	Esempio
`*`	qualsiasi elemento	`*`
`tag`	elementi con il nome del tag specificato	`div`
`ns|E`	elementi di tipo E nel namespace ns	`fb|name finds <fb:name> elements`
`#id`	elementi con ID attributo di "id"	`div#wrap, #logo`
`.class`	elementi con un nome di classe di "classe"	`div.left, .result`
`[attr]`	elementi con un attributo chiamato "attr" (con qualsiasi valore)	`a[href], [title]`
`[^attrPrefix]`	elementi con un nome di attributo che inizia con "attrPrefix". Utilizzare per trovare elementi con set di dati HTML5	`[^data-], div[^data-]`
`[attr=val]`	elementi con un attributo chiamato "attr" e valore uguale a "val"	`img[width=500], a[rel=nofollow]`
`[attr="val"]`	elementi con un attributo chiamato "attr" e valore uguale	`span[hello="Cleveland"][goodbye="Columbus"], a[rel="nofollow"]`

Modello	fiammiferi	Esempio
	a "val"	
`[attr^=valPrefix]`	elementi con un attributo chiamato "attr" e valore che inizia con "valPrefix"	`a[href^=http:]`
`[attr$=valSuffix]`	elementi con un attributo chiamato "attr" e valore che termina con "valSuffix"	`img[src$=.png]`
`[attr*=valContaining]`	elementi con un attributo chiamato "attr" e valore contenente "valContaining"	`a[href*=/search/]`
`[attr~=regex]`	elementi con un attributo chiamato "attr" e valore corrispondente all'espressione regolare	`img[src~=(?i)\.(png|jpe?g)]`
	Quanto sopra può essere combinato in qualsiasi ordine	`div.header[title]`

Selettore di riferimento completo

Examples

Selezione di elementi usando selettori CSS

```
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();
```

È possibile trovare una panoramica dettagliata di selettori supportati qui .

Estrai Twitter Markup

```
    // 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);
    }
```

Produzione

```
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 =
```

Leggi I selettori online: https://riptutorial.com/it/jsoup/topic/515/i-selettori

Capitolo 6: Web crawling con Jsoup

Examples

Estrazione di indirizzi email e collegamenti ad altre pagine

Jsoup può essere utilizzato per estrarre i link e l'indirizzo email da una pagina Web, quindi "bot del collector degli indirizzi email Web". In primo luogo, questo codice utilizza un'espressione regolare per estrarre gli indirizzi email e quindi utilizza i metodi forniti da Jsoup per estrarre gli URL dei collegamenti su la pagina.

```
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);

    }

}
```

Questo codice potrebbe anche essere facilmente esteso per visitare anche ricorsivamente quegli URL ed estrarre i dati dalle pagine collegate. Potrebbe anche essere facilmente utilizzato con una regex diversa per estrarre altri dati.

(Per favore non diventare uno spammer!)

Estrazione di dati JavaScript con Jsoup

In questo esempio, proveremo a trovare i dati JavaScript che contengono `backgroundColor:'#FFF'`. Quindi, cambieremo il valore di backgroundColor `'#FFF'` `'#ddd'`. Questo codice utilizza i `getWholeData()` e `setWholeData()` per manipolare i dati JavaScript. In alternativa, è possibile utilizzare il metodo `html()` per ottenere i dati di JavaScript.

```
// 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());
```

Produzione

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

Estrazione di tutti gli URL da un sito Web utilizzando JSoup (ricorsione)

In questo esempio verranno estratti tutti i collegamenti Web da un sito Web. Sto usando `http://stackoverflow.com/` per illustrazione. Qui viene utilizzata la ricorsione, in cui ogni pagina del collegamento ottenuta viene analizzata per la presenza di un `anchor tag` e tale collegamento viene nuovamente sottoposto alla stessa funzione.

La condizione `if(add && this_url.contains(my_site))` limiterà i risultati solo **al tuo dominio** .

```
    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) {

            }

        }
    }
```

Il programma impiegherà molto tempo a seconda del tuo sito web. Il codice sopra può essere esteso per estrarre dati (come titoli di pagine o testo o immagini) da un determinato sito web. Ti consiglierei di consultare i termini di utilizzo della società prima di scaricare il suo sito web.

Nell'esempio viene utilizzata la libreria JSoup per ottenere i collegamenti, è inoltre possibile ottenere i collegamenti utilizzando `your_url/sitemap.xml` .

Leggi Web crawling con Jsoup online: https://riptutorial.com/it/jsoup/topic/319/web-crawling-con-jsoup

Titoli di coda

S. No	Capitoli	Contributors
1	Iniziare con Jsoup	Alice, Community, Jeffrey Bosboom, JonasCz, Zack Teater
2	Accesso a siti Web con Jsoup	Joel Min, JonasCz, Stephan
3	Analisi delle pagine generate da Javascript	Zack Teater
4	Formattazione dell'output HTML	Zack Teater
5	I selettori	JonasCz, Stephan, still_learning, Zack Teater
6	Web crawling con Jsoup	Alice, JonasCz, r_D, RamenChef