eBook Gratuit

APPRENEZ Jsoup

eBook gratuit non affilié créé à partir des **contributeurs de Stack Overflow.**

#jsoup

Table des matières

À propos

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

Chapitre 1: Commencer avec Jsoup

Remarques

Jsoup est une bibliothèque d'analyse de données et d'extraction de données HTML pour Java, axée sur la flexibilité et la facilité d'utilisation. Il peut être utilisé pour extraire des données spécifiques des pages HTML, communément appelées "web scraping", ainsi que pour modifier le contenu des pages HTML et "nettoyer" le code HTML non approuvé avec une liste blanche de balises et d'attributs autorisés.

Support JavaScript

Jsoup ne prend pas en charge JavaScript et, de ce fait, aucun contenu généré dynamiquement ou ajouté à la page après le chargement de la page ne peut être extrait de la page. Si vous devez extraire du contenu qui est ajouté à la page avec JavaScript, il *existe* quelques options alternatives:

- Utilisez une bibliothèque qui prend en charge JavaScript, telle que Selenium, qui utilise un navigateur Web réel pour charger des pages, ou HtmlUnit.
- Reverse engineering de la façon dont la page charge ses données. En règle générale, les pages Web qui chargent des données dynamiquement le font via AJAX, et vous pouvez donc consulter l'onglet réseau des outils de développement de votre navigateur pour voir d'où proviennent les données, puis utiliser ces URL dans votre propre code. Voyez comment gratter les pages AJAX pour plus de détails.

Site officiel et documentation

Vous pouvez trouver diverses ressources liées à **Jsoup** sur **jsoup.org** , y compris Javadoc , des exemples d'utilisation dans le livre de recettes Jsoup et des téléchargements JAR . Consultez **le référentiel GitHub** pour le code source, les problèmes et les demandes d'extraction.

Télécharger

Jsoup est disponible sur Maven en tant que `org.jsoup.jsoup:jsoup` , Si vous utilisez Gradle (par exemple avec Android Studio), vous pouvez l'ajouter à votre projet en ajoutant ce qui suit à votre section de dépendances `build.gradle` :

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

Si vous utilisez Ant (Eclipse), ajoutez ce qui suit à votre section dépendances POM:

```
<dependency>
```

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

Jsoup est également disponible en tant **que JAR téléchargeable** pour d'autres environnements.

Versions

Version	Date de sortie
1.9.2	2016-05-17
1.8.3	2015-08-02

Examples

Extraire les URL et les titres des liens

Jsoup peut être utilisé pour extraire facilement tous les liens d'une page Web. Dans ce cas, nous pouvons utiliser Jsoup pour extraire uniquement les liens spécifiques que nous voulons, ici, ceux d'un en-tête `h3` sur une page. Nous pouvons également obtenir le texte des liens.

```
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();
}
```

Cela donne la sortie suivante:

```
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?

[...]
```

Qu'est-ce qu'il se passe ici:

- Tout d'abord, nous obtenons le document HTML à partir de l'URL spécifiée. Ce code définit également l'en-tête de l'agent utilisateur de la requête sur "Mozilla", de sorte que le site Web serve la page qu'il sert généralement aux navigateurs.
- Ensuite, utilisez `select(...)` et une boucle for pour obtenir tous les liens vers les questions

Stack Overflow, dans ce cas les liens ayant la classe `question-hyperlink` .

- Imprimez le texte de chaque lien avec `.text()` et le href du lien avec `attr("abs:href")` . Dans ce cas, nous utilisons `abs:` pour obtenir l'URL *absolue* , c.-à-d. avec le domaine et le protocole inclus.

Extraire l'URL complète du code HTML partiel

Sélectionner uniquement la valeur d'attribut d'un lien: href renvoie l'URL relative.

```
   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);
```

Sortie

```
/documentation
```

En transmettant l'URI de base dans la méthode d' `parse` et en utilisant la méthode `absUrl` au lieu de la méthode `attr` , nous pouvons extraire l'URL complète.

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

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

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

Sortie

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

Extraire les données du fichier de document HTML

Jsoup peut être utilisé pour manipuler ou extraire des données d'un **fichier** local contenant du code HTML. `filePath` est le chemin d'un fichier sur le disque. `ENCODING` est le nom du jeu de caractères souhaité, par exemple "Windows-31J". C'est facultatif.

```
 // 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");
```

Lire Commencer avec Jsoup en ligne: https://riptutorial.com/fr/jsoup/topic/297/commencer-avec-jsoup

Chapitre 2: Analyse des pages générées par Javascript

Examples

Analyse de la page générée par JavaScript avec Jsoup et HtmUnit

page.html - code source

```
<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 lorsqu'il est chargé dans le navigateur

Col1	Col2
0.0	0,1
1.0	1.1

Utiliser jsoup pour analyser page.html pour les données 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());
```

Sortie

(vide)

Qu'est-il arrivé?

Jsoup analyse le code source fourni par le serveur (ou dans ce cas chargé à partir du fichier). Il n'invoque pas les actions côté client telles que la manipulation JavaScript ou CSS DOM. Dans cet exemple, les lignes et les colonnes ne sont jamais ajoutées à la table de données.

Comment analyser ma page comme rendue dans le navigateur?

```
    // 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();
```

Sortie

```
0.0
0.1
1.0
1.1
```

Lire Analyse des pages générées par Javascript en ligne:
https://riptutorial.com/fr/jsoup/topic/4632/analyse-des-pages-generees-par-javascript

Chapitre 3: Formatage de la sortie HTML

Paramètres

Paramètre	Détail
`boolean outline()`	Obtenez si le mode contour est activé. La valeur par défaut est false. Si activé, les méthodes de sortie HTML considéreront toutes les balises comme un bloc.
`Document.OutputSettings outline(boolean)`	Activer ou désactiver le mode contour HTML.

Remarques

API Jsoup 1.9.2

Examples

Afficher tous les éléments en tant que bloc

Par défaut, Jsoup affichera uniquement les éléments de niveau bloc avec un saut de ligne final. Les éléments en ligne sont affichés sans saut de ligne.

Étant donné un fragment de corps, avec des éléments en ligne:

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

Impression avec Jsoup:

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

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

Résulte en:

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

Pour afficher la sortie avec chaque élément traité comme élément de bloc, l'option de `outline` doit être activée sur les `OutputSettings` de `OutputSettings` du document.

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

doc.outputSettings().outline(true);

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

Sortie

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

Source: JSoup - Formatage des éléments <option>

Lire Formatage de la sortie HTML en ligne: https://riptutorial.com/fr/jsoup/topic/5954/formatage-de-la-sortie-html

Chapitre 4: Se connecter à des sites Web avec Jsoup

Examples

Une simple demande d'authentification POST avec Jsoup

Une simple demande POST avec des données d'authentification est démontrée ci-dessous, notez que le champ `username` et `password` variera selon le site 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();
```

Une requête POST d'authentification plus complète avec Jsoup

La plupart des sites Web requièrent un processus beaucoup plus compliqué que celui démontré ci-dessus.

Les étapes courantes de connexion à un site Web sont les suivantes:

1. Obtenez le `cookie` unique `cookie` partir du formulaire de connexion initial.
2. Inspectez le formulaire de connexion pour voir quelle est l'URL de destination pour la demande d'authentification
3. Analyser le formulaire de connexion pour vérifier si `security token` doit être envoyé avec le nom d'utilisateur et le mot de passe.
4. Envoyer la demande

Vous trouverez ci-dessous un exemple de demande qui vous connectera au site Web de 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());
```

Journalisation avec FormElement

Dans cet exemple, nous allons nous connecter au site Web GitHub en utilisant 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);
    }
}
```

Toutes les données de formulaire sont gérées par la classe FormElement pour nous (même la détection de la méthode de formulaire). Une connexion prête à l'emploi est créée lors de l'appel de la méthode de soumission FormElement # . Tout ce que nous avons à faire est de compléter cette connexion avec des en-têtes supplémentaires (cookies, agent utilisateur, etc.) et de l'exécuter.

Lire Se connecter à des sites Web avec Jsoup en ligne:
https://riptutorial.com/fr/jsoup/topic/4631/se-connecter-a-des-sites-web-avec-jsoup

Chapitre 5: Sélecteurs

Remarques

Un sélecteur est une chaîne de sélecteurs simples, séparés par des combinateurs. Les sélecteurs ne sont pas sensibles à la casse (y compris les éléments, les attributs et les valeurs d'attribut).

Le sélecteur universel (*) est implicite quand aucun sélecteur d'élément n'est fourni (c.-à-d. Que * .header et .header sont équivalents).

Modèle	Allumettes	Exemple
`*`	n'importe quel élément	`*`
`tag`	éléments avec le nom de tag donné	`div`
`ns|E`	éléments de type E dans l'espace de noms ns	`fb|name finds <fb:name> elements`
`#id`	éléments avec identifiant d'attribut "id"	`div#wrap, #logo`
`.class`	éléments avec un nom de classe de "classe"	`div.left, .result`
`[attr]`	éléments avec un attribut nommé "attr" (avec n'importe quelle valeur)	`a[href], [title]`
`[^attrPrefix]`	éléments avec un nom d'attribut commençant par "attrPrefix". Utiliser pour rechercher des éléments avec des jeux de données HTML5	`[^data-], div[^data-]`
`[attr=val]`	éléments avec un attribut nommé "attr" et valeur égale à	`img[width=500], a[rel=nofollow]`

Modèle	Allumettes	Exemple
	"val"	
`[attr="val"]`	éléments avec un attribut nommé "attr" et valeur égale à "val"	`span[hello="Cleveland"][goodbye="Columbus"], a[rel="nofollow"]`
`[attr^=valPrefix]`	éléments avec un attribut nommé "attr", et valeur commençant par "valPrefix"	`a[href^=http:]`
`[attr$=valSuffix]`	éléments avec un attribut nommé "attr" et valeur se terminant par "valSuffix"	`img[src$=.png]`
`[attr*=valContaining]`	éléments avec un attribut nommé "attr" et valeur contenant "valContaining"	`a[href*=/search/]`
`[attr~=regex]`	éléments avec un attribut nommé "attr" et valeur correspondant à l'expression régulière	`img[src~=(?i)\.(png|jpe?g)]`
	Ce qui précède peut être combiné dans n'importe quel ordre	`div.header[title]`

Sélecteur référence complète

Examples

Sélection d'éléments à l'aide de sélecteurs 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();
```

Vous trouverez une vue d'ensemble détaillée des sélecteurs pris en charge ici .

Extraire le marquage Twitter

```
    // 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);
    }
```

Sortie

```
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 =
```

Lire Sélecteurs en ligne: https://riptutorial.com/fr/jsoup/topic/515/selecteurs

Chapitre 6: Web rampant avec Jsoup

Examples

Extraire des adresses email et des liens vers d'autres pages

Jsoup peut être utilisé pour extraire des liens et une adresse e-mail depuis une page Web, donc "bot collecteur d'adresse de messagerie Web" Tout d'abord, ce code utilise une expression régulière pour extraire les adresses e-mail, puis utilise les méthodes fournies par Jsoup la page.

```
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);

    }

}
```

Ce code pourrait également être facilement étendu pour visiter de manière récursive ces URL et extraire les données des pages liées. Il pourrait également être facilement utilisé avec une autre regex pour extraire d'autres données.

(S'il vous plaît ne devenez pas un spammeur!)

Extraire des données JavaScript avec Jsoup

Dans cet exemple, nous allons essayer de trouver des données JavaScript contenant `backgroundColor:'#FFF'` . Ensuite, nous changerons la valeur de backgroundColor `'#FFF' '#ddd'` . Ce code utilise les `getWholeData()` et `setWholeData()` pour manipuler les données JavaScript. La méthode `html()` peut également être utilisée pour obtenir des données 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());
```

Sortie

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

Extraire toutes les URL d'un site Web en utilisant JSoup (récursivité)

Dans cet exemple, nous allons extraire tous les liens Web d'un site Web. J'utilise `http://stackoverflow.com/` pour illustration. Ici, la récursivité est utilisée, où la page de chaque lien obtenu est analysée pour la présence d'une `anchor tag` et ce lien est à nouveau soumis à la même fonction.

La condition `if(add && this_url.contains(my_site))` limitera les résultats **à votre domaine uniquement** .

```
    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) {

            }

        }
    }
```

Le programme prendra beaucoup de temps à exécuter en fonction de votre site Web. Le code ci-dessus peut être étendu pour extraire des données (comme des titres de pages ou du texte ou des images) d'un site Web particulier. Je vous recommande de consulter les conditions d'utilisation de la société avant de créer son site Web.

L'exemple utilise la bibliothèque JSoup pour obtenir les liens, vous pouvez également obtenir les liens en utilisant `your_url/sitemap.xml` .

Lire Web rampant avec Jsoup en ligne: https://riptutorial.com/fr/jsoup/topic/319/web-rampant-avec-jsoup

Crédits

S. No	Chapitres	Contributeurs
1	Commencer avec Jsoup	Alice, Community, Jeffrey Bosboom, JonasCz, Zack Teater
2	Analyse des pages générées par Javascript	Zack Teater
3	Formatage de la sortie HTML	Zack Teater
4	Se connecter à des sites Web avec Jsoup	Joel Min, JonasCz, Stephan
5	Sélecteurs	JonasCz, Stephan, still_learning, Zack Teater
6	Web rampant avec Jsoup	Alice, JonasCz, r_D, RamenChef