Бесплатная электронная книга

УЧУСЬ

Jsoup

Free unaffiliated eBook created from
Stack Overflow contributors.

#jsoup

Около

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

глава 1: Начало работы с Jsoup

замечания

Jsoup - это библиотека анализа и анализа данных HTML для Java, ориентированная на гибкость и простоту использования. Его можно использовать для извлечения отдельных данных с HTML-страниц, которые обычно называются «веб-скребками», а также для изменения содержимого страниц HTML и «чистого» ненадежного HTML с помощью белого списка разрешенных тегов и атрибутов.

Поддержка JavaScript

Jsoup не поддерживает JavaScript , и из-за этого из страницы невозможно извлечь любой динамически созданный контент или контент, который добавляется на страницу после загрузки страницы. Если вам нужно извлечь содержимое , которое добавляется к странице с JavaScript, существует несколько альтернативных вариантов:

- Используйте библиотеку, которая поддерживает JavaScript, например Selenium, которая использует фактический веб-браузер для загрузки страниц или HtmlUnit.
- Обратный инженер, как страница загружает данные. Как правило, веб-страницы, которые динамически загружают данные с помощью AJAX, таким образом, вы можете посмотреть вкладку сети инструментов разработчика вашего браузера, чтобы узнать, откуда загружаются данные, а затем использовать эти URL в своем собственном коде. Узнайте, как очистить страницы AJAX для получения более подробной информации.

Официальный сайт и документация

Вы можете найти различные ресурсы **Jsoup** на **jsoup.org** , включая Javadoc , примеры использования в кулинарной книге Jsoup и загрузке JAR . См. **Репозиторий GitHub** для исходного кода, проблем и запросов на вытягивание.

Скачать

Jsoup доступен на Maven как `org.jsoup.jsoup:jsoup` Если вы используете Gradle (например, с Android Studio), вы можете добавить его в свой проект, добавив следующее в свой `build.gradle` зависимостей `build.gradle` :

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

Если вы используете Ant (Eclipse), добавьте следующее в раздел зависимостей POM:

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

Jsoup также доступен в качестве **загружаемого JAR** для других сред.

Версии

Версия	Дата выхода
1.9.2	2016-05-17
1.8.3	2015-08-02

Examples

Извлеките URL-адреса и заголовки ссылок

Jsoup можно использовать для простого извлечения всех ссылок с веб-страницы. В этом случае мы можем использовать Jsoup для извлечения только определенных ссылок, которые мы хотим, здесь, в заголовке `h3` на странице. Мы также можем получить текст ссылок.

```
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();
}
```

Это дает следующий результат:

```
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?

[...]
```

Что тут происходит:

- Во-первых, мы получаем HTML-документ с указанным URL-адресом. Этот код также устанавливает заголовок User Agent запроса в «Mozilla», так что веб-сайт служит странице, которая обычно будет использоваться для браузеров.
- Затем используйте `select(...)` и цикл for, чтобы получить все ссылки на вопросы переполнения стека, в этом случае ссылки, которые имеют класс `question-hyperlink` .
- Распечатайте текст каждой ссылки с помощью `.text()` и href ссылки с `attr("abs:href")` . В этом случае мы используем `abs:` для получения *абсолютного* URL, т. Е. с включенным доменом и протоколом.

Извлечь полный URL из частичного HTML

Выбрав только значение атрибута ссылки: href вернет относительный URL.

```
   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);
```

Выход

```
/documentation
```

`absUrl` базовый URI в метод `parse` и используя метод `absUrl` вместо `attr` , мы можем извлечь полный URL.

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

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

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

Выход

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

Извлечение данных из файла документа HTML

Jsoup можно использовать для манипулирования или извлечения данных из локального **файла** , содержащего HTML. `filePath` - это путь к файлу на диске. `ENCODING` например, « Windows-31J». Это необязательно.

```
// 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");
```

Прочитайте Начало работы с Jsoup онлайн: https://riptutorial.com/ru/jsoup/topic/297/начало-работы-с-jsoup

глава 2: Веб-сканирование с помощью Jsoup

Examples

Извлечение адресов электронной почты и ссылок на другие страницы

Jsoup можно использовать для извлечения ссылок и адреса электронной почты с веб-страницы, таким образом, «бот-коллектор веб-адреса электронной почты». Сначала этот код использует регулярное выражение для извлечения адресов электронной почты, а затем использует методы, предоставленные Jsoup, для извлечения URL-адресов ссылок на страница.

```
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);

    }

}
```

Этот код также можно легко расширить, чтобы также рекурсивно посетить эти URL-адреса и извлечь данные со связанных страниц. Он также может быть легко использован с другим регулярным выражением для извлечения других данных.

(Пожалуйста, не становитесь спамером!)

Извлечение данных JavaScript с помощью Jsoup

В этом примере мы попытаемся найти данные JavaScript, содержащие `backgroundColor:'#FFF'`

. Затем мы изменим значение backgroundColor `'#FFF' '#ddd'` . Этот код использует `getWholeData()` и `setWholeData()` для управления данными JavaScript. Кроме того, метод `html()` может использоваться для получения данных 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());
```

Выход

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

Извлечение всех URL-адресов с веб-сайта с использованием JSoup (рекурсия)

В этом примере мы будем извлекать все веб-ссылки с веб-сайта. Я использую `http://stackoverflow.com/` для иллюстрации. Здесь используется рекурсия, где каждая страница получаемой ссылки анализируется на наличие `anchor tag` и эта ссылка снова представляется в ту же функцию.

Условие `if(add && this_url.contains(my_site))` ограничит результаты **только вашим доменом** .

```
    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) {

            }

        }
    }
```

В зависимости от вашего сайта программа займет много времени. Вышеприведенный код может быть расширен для извлечения данных (например, заголовков страниц или текста или изображений) с определенного веб-сайта. Я бы порекомендовал вам ознакомиться с условиями использования компании до того, как нащупал ее сайт.

В примере используется библиотека JSoup для получения ссылок, вы также можете получить ссылки, используя `your_url/sitemap.xml` .

Прочитайте Веб-сканирование с помощью Jsoup онлайн:
https://riptutorial.com/ru/jsoup/topic/319/веб-сканирование-с-помощью-jsoup

глава 3: Вход на сайты с Jsoup

Examples

Простой аутентификационный запрос POST с Jsoup

Простой запрос POST с данными аутентификации показан ниже, обратите внимание, что поле `username` и `password` будет зависеть от веб-сайта:

```
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();
```

Более полный запрос аутентификации POST с Jsoup

Для большинства веб-сайтов требуется гораздо более сложный процесс, чем описанный выше.

Общие шаги для входа на веб-сайт:

1. Получите уникальный `cookie` из начальной формы входа.
2. Проверьте форму входа в систему, чтобы узнать, что URL-адрес назначения для запроса аутентификации
3. Разберите форму входа, чтобы проверить наличие `security token` который необходимо отправить вместе с именем пользователя и паролем.
4. Отправьте запрос.

Ниже приведен пример запроса, который будет регистрировать вас на веб-сайте 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());
```

Вход в систему с помощью FormElement

В этом примере мы запишемся на сайт GitHub , используя класс 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);
    }
}
```

Все данные формы обрабатываются классом FormElement для нас (даже при обнаружении метода формы). Готовое соединение создается при вызове метода отправки FormElement # . Все, что нам нужно сделать, это завершить соединение с дополнительными заголовками (куки, пользовательский агент и т. Д.) И выполнить его.

Прочитайте Вход на сайты с Jsoup онлайн: https://riptutorial.com/ru/jsoup/topic/4631/вход-на-сайты-с-jsoup

глава 4: Разбор Javascript Сгенерированные страницы

Examples

Анализ JavaScript сгенерированной страницы с помощью Jsoup и HtmUnit

page.html - исходный код

```
<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 при загрузке в браузер

Col1	Col2
0.0	0,1
1,0	1,1

Использование jsoup для разбора page.html для данных 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());
```

Выход

(Пусто)

Что случилось?

Jsoup анализирует исходный код с доставкой с сервера (или в этом случае загружается из файла). Он не вызывает действия на стороне клиента, такие как JavaScript или CSS DOM-манипуляция. В этом примере строки и столбцы никогда не присоединяются к таблице данных.

Как анализировать мою страницу как отображаемую в браузере?

```
    // 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();
```

Выход

```
0.0
0.1
1.0
1.1
```

Прочитайте Разбор Javascript Сгенерированные страницы онлайн:
https://riptutorial.com/ru/jsoup/topic/4632/разбор-javascript-сгенерированные-страницы

глава 5: Селекторы

замечания

Селектор представляет собой цепочку простых селекторов, разделенных комбинаторами. Селекторы не чувствительны к регистру (включая элементы, атрибуты и значения атрибутов).

Универсальный селектор (*) является неявным, если не задан селектор элементов (т.е. * .header и .header эквивалентны).

Шаблон	Матчи	пример
`*`	любой элемент	`*`
`tag`	элементы с указанным именем тега	`div`
`ns|E`	элементы типа E в пространстве имен ns	`fb|name finds <fb:name> elements`
`#id`	элементы с идентификатором атрибута "id"	`div#wrap, #logo`
`.class`	элементы с именем класса "class"	`div.left, .result`
`[attr]`	элементы с атрибутом с именем «attr» (с любым значением)	`a[href], [title]`
`[^attrPrefix]`	элементы с именем атрибута, начинающимся с « attrPrefix». Используйте для поиска элементов с наборами данных HTML5	`[^data-], div[^data-]`

Шаблон	Матчи	пример
`[attr=val]`	элементы с атрибутом с именем «attr» и значением, равным «val»,	`img[width=500], a[rel=nofollow]`
`[attr="val"]`	элементы с атрибутом с именем «attr» и значением, равным «val»,	`span[hello="Cleveland"][goodbye="Columbus"], a[rel="nofollow"]`
`[attr^=valPrefix]`	элементы с атрибутом с именем «attr» и значением, начинающимся с « valPrefix»,	`a[href^=http:]`
`[attr$=valSuffix]`	элементы с атрибутом с именем «attr», а значение, заканчивающееся на «valSuffix»,	`img[src$=.png]`
`[attr*=valContaining]`	элементы с атрибутом с именем "attr" и значением, содержащим " valContaining"	`a[href*=/search/]`
`[attr~=regex]`	элементы с атрибутом с именем «attr» и значением, соответствующим регулярному выражению	`img[src~=(?i)\.(png|jpe?g)]`
	Вышеупомянутые могут быть объединены в любом порядке	`div.header[title]`

Полностью ссылка Selector

Examples

Выбор элементов с помощью селекторов 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();
```

Вы можете найти подробный обзор поддерживаемых селекторов здесь .

Извлечь разметку 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);
    }
```

Выход

```
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 =
```

Прочитайте Селекторы онлайн: https://riptutorial.com/ru/jsoup/topic/515/селекторы

глава 6: Форматирование вывода HTML

параметры

параметр	подробность
`boolean outline()`	Получить, если включен режим контура. Значение по умолчанию - false. Если включено, методы вывода HTML будут рассматривать все теги как блок.
`Document.OutputSettings outline(boolean)`	Включить или отключить режим контуров HTML.

замечания

API Jsoup 1.9.2

Examples

Отображение всех элементов в виде блока

По умолчанию Jsoup будет отображать только элементы уровня блока с прерыванием строки. Встроенные элементы отображаются без разрыва строки.

Учитывая фрагмент тела, с встроенными элементами:

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

Печать с помощью Jsoup:

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

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

Результаты в:

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

Чтобы отобразить вывод с каждым элементом, рассматриваемым как элемент блока, параметр `outline` должен быть включен в `OutputSettings` документа.

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

doc.outputSettings().outline(true);

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

Выход

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

Источник: JSoup - форматирование элементов <option>

Прочитайте Форматирование вывода HTML онлайн: https://riptutorial.com/ru/jsoup/topic/5954/форматирование-вывода-html

кредиты

S. No	Главы	Contributors
1	Начало работы с Jsoup	Alice, Community, Jeffrey Bosboom, JonasCz, Zack Teater
2	Веб-сканирование с помощью Jsoup	Alice, JonasCz, r_D, RamenChef
3	Вход на сайты с Jsoup	Joel Min, JonasCz, Stephan
4	Разбор Javascript Сгенерированные страницы	Zack Teater
5	Селекторы	JonasCz, Stephan, still_learning, Zack Teater
6	Форматирование вывода HTML	Zack Teater