EBook Gratis

APRENDIZAJE Jsoup

Free unaffiliated eBook created from
Stack Overflow contributors.

#jsoup

Tabla de contenido

Acerca de ... 1

Capítulo 1: Empezando con Jsoup ... 2

Observaciones ... 2

Soporte de JavaScript ... 2

Página oficial y documentación ... 2

Descargar ... 2

Versiones ... 3

Examples ... 3

Extrae las URL y títulos de los enlaces ... 3

Extraer URL completa de HTML parcial ... 4

Extraer los datos del archivo de documento HTML ... 4

Capítulo 2: Análisis de páginas generadas en Javascript ... 6

Examples ... 6

Análisis de la página generada por JavaScript con Jsoup y HtmUnit ... 6

Capítulo 3: Formato de salida HTML ... 8

Parámetros ... 8

Observaciones ... 8

Examples ... 8

Mostrar todos los elementos como bloque ... 8

Capítulo 4: Iniciar sesión en sitios web con Jsoup ... 10

Examples ... 10

Una simple solicitud POST de autenticación con Jsoup ... 10

Una solicitud POST de autenticación más completa con Jsoup ... 10

Registro con FormElement ... 11

Capítulo 5: Selectores ... 13

Observaciones ... 13

Examples ... 14

Selección de elementos mediante selectores de CSS ... 14

Extraer el marcado de Twitter ... 15

Capítulo 6: Web crawling con jsoup ... 17

Examples 17
Extraer direcciones de correo electrónico y enlaces a otras páginas 17
Extraer datos de JavaScript con Jsoup 17
Extraer todas las URL de un sitio web usando JSoup (recursión) 18
Creditos 20

Acerca de

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

Capítulo 1: Empezando con Jsoup

Observaciones

Jsoup es una biblioteca de análisis de HTML y extracción de datos para Java, enfocada en la flexibilidad y la facilidad de uso. Se puede usar para extraer datos específicos de páginas HTML, lo que comúnmente se conoce como "web scraping", así como modificar el contenido de las páginas HTML y "limpiar" HTML no confiable con una lista blanca de etiquetas y atributos permitidos.

Soporte de JavaScript

Jsoup no admite JavaScript y, debido a esto, cualquier contenido generado dinámicamente o que se agregue a la página después de la carga de la página no se puede extraer de la página. Si necesita extraer el contenido que se añade a la página con JavaScript, *hay* algunas opciones alternativas:

- Use una biblioteca que admita JavaScript, como Selenium, que usa un navegador web real para cargar páginas, o HtmlUnit.
- Ingeniero inverso cómo la página carga sus datos. Normalmente, las páginas web que cargan datos dinámicamente lo hacen a través de AJAX y, por lo tanto, puede consultar la pestaña de red de las herramientas de desarrollo de su navegador para ver desde dónde se están cargando los datos y luego usar esas URL en su propio código. Vea cómo raspar las páginas AJAX para más detalles.

Página oficial y documentación.

Puede encontrar varios recursos relacionados con **Jsoup** en **jsoup.org** , incluido Javadoc , ejemplos de uso en el libro de cocina de Jsoup y descargas de JAR . Consulte **el repositorio de GitHub** para ver el código fuente, los problemas y las solicitudes de extracción.

Descargar

Jsoup está disponible en Maven como `org.jsoup.jsoup:jsoup` , si está utilizando Gradle (por ejemplo, con Android Studio), puede agregarlo a su proyecto agregando lo siguiente a su sección de dependencias de `build.gradle` :

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

Si está usando Ant (Eclipse), agregue lo siguiente a la sección de dependencias de su POM:

```
<dependency>
```

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

Jsoup también está disponible como **JAR descargable** para otros entornos.

Versiones

Versión	Fecha de lanzamiento
1.9.2	2016-05-17
1.8.3	2015-08-02

Examples

Extrae las URL y títulos de los enlaces.

Jsoup se puede utilizar para extraer fácilmente todos los enlaces de una página web. En este caso, podemos usar Jsoup para extraer solo los enlaces específicos que queremos, aquí, los de un encabezado `h3` en una página. También podemos obtener el texto de los enlaces.

```
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();
}
```

Esto da el siguiente resultado:

```
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?

[...]
```

Que esta pasando aqui:

- Primero, obtenemos el documento HTML de la URL especificada. Este código también establece el encabezado del Agente de usuario de la solicitud en "Mozilla", de modo que el sitio web sirva la página que usualmente serviría a los navegadores.
- Luego, use `select(...)` y un bucle for para obtener todos los enlaces a las preguntas de

desbordamiento de pila, en este caso los enlaces que tienen el `question-hyperlink` clase.

- Imprima el texto de cada enlace con `.text()` y el href del enlace con `attr("abs:href")` . En este caso, usamos `abs:` para obtener la URL *absoluta* , es decir. Con el dominio y protocolo incluidos.

Extraer URL completa de HTML parcial

Seleccionando solo el valor del atributo de un enlace: href devolverá la URL relativa.

```
   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);
```

Salida

```
/documentation
```

Al pasar el URI base al método de `parse` y usar el método `absUrl` lugar de `attr` , podemos extraer la URL completa.

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

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

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

Salida

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

Extraer los datos del archivo de documento HTML

Jsoup se puede usar para manipular o extraer datos de un **archivo** en local que contiene HTML. `filePath` es la ruta de un archivo en el disco. `ENCODING` se desea Juego de caracteres Nombre por ejemplo "Windows-31J". Es opcional.

```
 // 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");
```

Lea Empezando con Jsoup en línea: https://riptutorial.com/es/jsoup/topic/297/empezando-con-jsoup

Capítulo 2: Análisis de páginas generadas en Javascript

Examples

Análisis de la página generada por JavaScript con Jsoup y HtmUnit

page.html - código fuente

```
<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 cuando se carga en el navegador

Col1	Col2
0.0	0.1
1.0	1.1

Usando jsoup para analizar page.html para datos 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());
```

Salida

(vacío)

¿Que pasó?

Jsoup analiza el código fuente como se entrega desde el servidor (o en este caso, cargado desde un archivo). No invoca acciones del lado del cliente, como la manipulación de JavaScript o CSS DOM. En este ejemplo, las filas y columnas nunca se agregan a la tabla de datos.

¿Cómo analizar mi página como se muestra en el navegador?

```
    // 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();
```

Salida

```
0.0
0.1
1.0
1.1
```

Lea Análisis de páginas generadas en Javascript en línea:
https://riptutorial.com/es/jsoup/topic/4632/analisis-de-paginas-generadas-en-javascript

Capítulo 3: Formato de salida HTML

Parámetros

Parámetro	Detalle
`boolean outline()`	Obtener si el modo de esquema está habilitado. El valor predeterminado es falso. Si está habilitado, los métodos de salida HTML considerarán todas las etiquetas como bloque.
`Document.OutputSettings outline(boolean)`	Habilitar o deshabilitar el modo de esquema HTML.

Observaciones

Jsoup 1.9.2 API

Examples

Mostrar todos los elementos como bloque

De forma predeterminada, Jsoup mostrará solo los elementos de nivel de bloque con un salto de línea posterior. Los elementos en línea se muestran sin un salto de línea.

Dado un fragmento de cuerpo, con elementos en línea:

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

Imprimiendo con Jsoup:

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

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

Resultados en:

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

Para mostrar la salida con cada elemento tratado como un elemento de bloque, la opción de `outline` debe estar habilitada en los `OutputSettings` de `OutputSettings` del documento.

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

doc.outputSettings().outline(true);

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

Salida

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

Fuente: JSoup - Formateo de los elementos <option>

Lea Formato de salida HTML en línea: https://riptutorial.com/es/jsoup/topic/5954/formato-de-salida-html

Capítulo 4: Iniciar sesión en sitios web con Jsoup

Examples

Una simple solicitud POST de autenticación con Jsoup

A continuación se muestra una solicitud POST simple con datos de autenticación, tenga en cuenta que el campo de `username` y `password` variará según el sitio 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 solicitud POST de autenticación más completa con Jsoup

La mayoría de los sitios web requieren un proceso mucho más complicado que el demostrado anteriormente.

Los pasos comunes para iniciar sesión en un sitio web son:

1. Obtenga la `cookie` única del formulario de inicio de sesión inicial.
2. Inspeccione el formulario de inicio de sesión para ver cuál es la URL de destino para la solicitud de autenticación
3. Analice el formulario de inicio de sesión para verificar cualquier `security token` que deba enviarse junto con el nombre de usuario y la contraseña.
4. Enviar la solicitud.

A continuación se muestra un ejemplo de solicitud que lo conectará con el sitio 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());
```

Registro con FormElement

En este ejemplo, iniciaremos sesión en el sitio web de GitHub utilizando la clase 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);
    }
}
```

Todos los datos del formulario son manejados por la clase FormElement para nosotros (incluso la detección del método del formulario). Una conexión ya hecha se crea cuando se invoca el método de envío FormElement # . Todo lo que tenemos que hacer es completar esta conexión con encabezados adicionales (cookies, usuario-agente, etc.) y ejecutarla.

Lea Iniciar sesión en sitios web con Jsoup en línea:
https://riptutorial.com/es/jsoup/topic/4631/iniciar-sesion-en-sitios-web-con-jsoup

Capítulo 5: Selectores

Observaciones

Un selector es una cadena de selectores simples, separados por combinadores. Los selectores no distinguen entre mayúsculas y minúsculas (incluidos los elementos, los atributos y los valores de los atributos).

El selector universal (*) está implícito cuando no se suministra ningún selector de elementos (es decir, * .header y .header son equivalentes).

Modelo	Partidos	Ejemplo
`*`	cualquier elemento	`*`
`tag`	elementos con el nombre de etiqueta dado	`div`
`ns|E`	Elementos de tipo E en el espacio de nombres ns.	`fb|name finds <fb:name> elements`
`#id`	elementos con ID de atributo de "id"	`div#wrap, #logo`
`.class`	elementos con un nombre de clase de "clase"	`div.left, .result`
`[attr]`	Elementos con un atributo llamado "attr" (con cualquier valor)	`a[href], [title]`
`[^attrPrefix]`	elementos con un nombre de atributo que comience con "attrPrefix". Úsalo para encontrar elementos con datasets HTML5	`[^data-], div[^data-]`
`[attr=val]`	Elementos con un atributo llamado "attr" y valor igual a "val"	`img[width=500], a[rel=nofollow]`
`[attr="val"]`	Elementos con un	`span[hello="Cleveland"][goodbye="Columbus"],`

Modelo	Partidos	Ejemplo
	atributo llamado "attr" y valor igual a "val"	`a[rel="nofollow"]`
`[attr^=valPrefix]`	elementos con un atributo llamado "attr" y valor que comienza con "valPrefix"	`a[href^=http:]`
`[attr$=valSuffix]`	elementos con un atributo llamado "attr", y el valor termina con "valSuffix"	`img[src$=.png]`
`[attr*=valContaining]`	Elementos con un atributo llamado "attr" y un valor que contiene "valContaining"	`a[href*=/search/]`
`[attr~=regex]`	Elementos con un atributo llamado "attr" y un valor que coincide con la expresión regular	`img[src~=(?i)\.(png|jpe?g)]`
	Lo anterior se puede combinar en cualquier orden.	`div.header[title]`

Selector de referencia completa

Examples

Selección de elementos mediante selectores de 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();
```

Puede encontrar una descripción detallada de los selectores admitidos aquí .

Extraer el marcado de 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);
    }
```

Salida

```
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 =
```

Lea Selectores en línea: https://riptutorial.com/es/jsoup/topic/515/selectores

Capítulo 6: Web crawling con jsoup

Examples

Extraer direcciones de correo electrónico y enlaces a otras páginas.

Jsoup se puede usar para extraer enlaces y direcciones de correo electrónico de una página web, por lo tanto, "bot de recopilador de direcciones de correo electrónico web" Primero, este código usa una expresión regular para extraer las direcciones de correo electrónico, y luego utiliza los métodos proporcionados por Jsoup para extraer las URL de los enlaces. la página.

```
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);

    }

}
```

Este código también se puede extender fácilmente para visitar recursivamente esas URL y extraer datos de páginas vinculadas. También se podría usar fácilmente con una expresión regular diferente para extraer otros datos.

(Por favor, no te conviertas en un spammer!)

Extraer datos de JavaScript con Jsoup

En este ejemplo, intentaremos encontrar datos de JavaScript que contengan `backgroundColor:'#FFF'` . Luego, cambiaremos el valor de backgroundColor `'#FFF' '#ddd'` . Este código utiliza los `getWholeData()` y `setWholeData()` para manipular los datos de JavaScript. Alternativamente, el método `html()` se puede usar para obtener datos de 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());
```

Salida

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

Extraer todas las URL de un sitio web usando JSoup (recursión)

En este ejemplo extraeremos todos los enlaces web de un sitio web. Estoy utilizando `http://stackoverflow.com/` para ilustración. Aquí se utiliza la recursión, donde se analiza la página de cada enlace obtenido para detectar la presencia de una `anchor tag` y ese enlace se envía nuevamente a la misma función.

La condición `if(add && this_url.contains(my_site))` limitará los resultados solo **a su 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) {

            }

        }
    }
```

El programa tardará mucho tiempo en ejecutarse dependiendo de su sitio web. El código anterior se puede ampliar para extraer datos (como títulos de páginas o texto o imágenes) de un sitio web en particular. Le recomendaría que revise los términos de uso de la compañía antes de abrir su sitio web.

El ejemplo usa la biblioteca JSoup para obtener los enlaces, también puede obtener los enlaces usando `your_url/sitemap.xml` .

Lea Web crawling con jsoup en línea: https://riptutorial.com/es/jsoup/topic/319/web-crawling-con-jsoup

Creditos

S. No	Capítulos	Contributors
1	Empezando con Jsoup	Alice, Community, Jeffrey Bosboom, JonasCz, Zack Teater
2	Análisis de páginas generadas en Javascript	Zack Teater
3	Formato de salida HTML	Zack Teater
4	Iniciar sesión en sitios web con Jsoup	Joel Min, JonasCz, Stephan
5	Selectores	JonasCz, Stephan, still_learning, Zack Teater
6	Web crawling con jsoup	Alice, JonasCz, r_D, RamenChef