jsoup Cookbook(中文版)

入门

1. [解析和遍历一个html文档](#_1.解析和遍历一个HTML文档)

输入

1. [解析一个html字符串](#_2.解析一个HTML字符串)
2. [解析一个body片断](#_3.解析一个body片断)
3. [根据一个url加载Document对象](#_4.从一个URL加载一个Document)
4. [根据一个文件加载Document对象](#_5.从一个文件加载一个文档)

数据抽取

1. [使用dom方法来遍历一个Document对象](#_6.使用DOM方法来遍历一个文档)
2. [使用选择器语法来查找元素](#_7.使用选择器语法来查找元素)
3. [从元素集合抽取属性、文本和html内容](#_8.从元素抽取属性，文本和HTML)
4. [URL处理](#_9.处理URLs)
5. [程序示例：获取所有链接](#_10.示例程序:_获取所有链接)

数据修改

1. [设置属性值](#_11.设置属性的值)
2. [设置元素的html内容](#_12.设置一个元素的HTML内容)
3. [设置元素的文本内容](#_13.设置元素的文本内容)

html清理

1. [消除不受信任的html (来防止xss攻击)](#_14.消除不受信任的HTML_(来防止XSS攻击))

# 1.解析和遍历一个HTML文档

如何解析一个HTML文档：

String html = "<html><head><title>First parse</title></head>"

+ "<body><p>Parsed HTML into a doc.</p></body></html>";

Document doc = Jsoup.parse(html);

(更详细内容可查看 [解析一个HTML字符串](http://jsoup.org/cookbook/input/parse-document-from-string).)

其解析器能够尽最大可能从你提供的HTML文档来创见一个干净的解析结果，无论HTML的格式是否完整。比如它可以处理：

* 没有关闭的标签 (比如： <p>Lorem <p>Ipsum parses to <p>Lorem</p> <p>Ipsum</p>)
* 隐式标签 (比如. 它可以自动将 <td>Table data</td>包装成<table><tr><td>?)
* 创建可靠的文档结构（html标签包含head 和 body，在head只出现恰当的元素）

## 一个文档的对象模型

* 文档由多个Elements和TextNodes组成 (以及其它辅助nodes：详细可查看：[nodes package tree](http://jsoup.org/apidocs/org/jsoup/nodes/package-tree.html)).
* 其继承结构如下：[Document](http://jsoup.org/apidocs/org/jsoup/nodes/Document.html" \o "A HTML Document.)继承[Element](http://jsoup.org/apidocs/org/jsoup/nodes/Element.html" \o "A HTML element consists of a tag name, attributes, and child nodes (including text nodes and other elements).)继承[Node](http://jsoup.org/apidocs/org/jsoup/nodes/Node.html" \o "The base, abstract Node model.). [TextNode](http://jsoup.org/apidocs/org/jsoup/nodes/TextNode.html" \o "A text node.)继承 [Node](http://jsoup.org/apidocs/org/jsoup/nodes/Node.html" \o "The base, abstract Node model.).
* 一个Element包含一个子节点集合，并拥有一个父Element。他们还提供了一个唯一的子元素过滤列表。

## 参见

* 数据抽取：[DOM遍历](http://jsoup.org/cookbook/extracting-data/dom-navigation)
* 数据抽取：[Selector syntax](http://jsoup.org/cookbook/extracting-data/selector-syntax)

# 2.解析一个HTML字符串

## 存在问题

来自用户输入，一个文件或一个网站的HTML字符串，你可能需要对它进行解析并取其内容，或校验其格式是否完整，或想修改它。怎么办？jsonu能够帮你轻松解决这些问题

## 解决方法

使用静态[Jsoup.parse(String html)](http://jsoup.org/apidocs/org/jsoup/Jsoup.html" \l "parse(java.lang.String)" \o "Parse HTML into a Document.) 方法或 [Jsoup.parse(String html, String baseUri)](http://jsoup.org/apidocs/org/jsoup/Jsoup.html" \l "parse(java.lang.String, java.lang.String)" \o "Parse HTML into a Document.)示例代码：

String html = "<html><head><title>First parse</title></head>"

+ "<body><p>Parsed HTML into a doc.</p></body></html>";

Document doc = Jsoup.parse(html);

## 描述

[parse(String html, String baseUri)](http://jsoup.org/apidocs/org/jsoup/parser/Parser.html" \l "parse(java.lang.String, java.lang.String)" \o "Parse HTML into a Document.) 这方法能够将输入的HTML解析为一个新的文档 (Document），参数 baseUri 是用来将相对 URL 转成绝对URL，并指定从哪个网站获取文档。如这个方法不适用，你可以使用 [parse(String html)](http://jsoup.org/apidocs/org/jsoup/Jsoup.html" \l "parse(java.lang.String)" \o "Parse HTML into a Document.) 方法来解析成HTML字符串如上面的示例。.

只要解析的不是空字符串，就能返回一个结构合理的文档，其中包含(至少) 一个head和一个body元素。

一旦拥有了一个Document，你就可以使用Document中适当的方法或它父类 [Element](http://jsoup.org/apidocs/org/jsoup/nodes/Element.html" \o "A HTML element consists of a tag name, attributes, and child nodes (including text nodes and other elements).)和[Node](http://jsoup.org/apidocs/org/jsoup/nodes/Node.html" \o "The base, abstract Node model.)中的方法来取得相关数据。

# 3.解析一个body片断

## 问题

假如你有一个HTML片断 (比如. 一个 div 包含一对 p 标签; 一个不完整的HTML文档) 想对它进行解析。这个HTML片断可以是用户提交的一条评论或在一个CMS页面中编辑body部分。

## 办法

使用[Jsoup.parseBodyFragment(String html)](http://jsoup.org/apidocs/org/jsoup/Jsoup.html" \l "parseBodyFragment(java.lang.String)" \o "Parse a fragment of HTML, with the assumption that it forms the body of the HTML.)方法.

String html = "<div><p>Lorem ipsum.</p>";

Document doc = Jsoup.parseBodyFragment(html);

Element body = doc.body();

## 说明

parseBodyFragment 方法创建一个空壳的文档，并插入解析过的HTML到body元素中。假如你使用正常的 [Jsoup.parse(String html)](http://jsoup.org/apidocs/org/jsoup/Jsoup.html" \l "parse(java.lang.String)" \o "Parse HTML into a Document.) 方法，通常你也可以得到相同的结果，但是明确将用户输入作为 body片段处理，以确保用户所提供的任何糟糕的HTML都将被解析成body元素。

[Document.body()](http://jsoup.org/apidocs/org/jsoup/nodes/Document.html" \l "body()" \o "Accessor to the document's body element.) 方法能够取得文档body元素的所有子元素，与 doc.getElementsByTag("body")相同。

## 保证安全Stay safe

假如你可以让用户输入HTML内容，那么要小心避免跨站脚本攻击。利用基于 [Whitelist](http://jsoup.org/apidocs/org/jsoup/safety/Whitelist.html" \o "Whitelists define what HTML (elements and attributes) to allow through the cleaner.) 的清除器和 [clean(String bodyHtml, Whitelist whitelist)](http://jsoup.org/apidocs/org/jsoup/Jsoup.html" \l "clean(java.lang.String, org.jsoup.safety.Whitelist)" \o "Get safe HTML from untrusted input HTML, by parsing input HTML and filtering it through a white-list of permitted tags and attributes.)方法来清除用户输入的恶意内容。

# 4.从一个URL加载一个Document

## 存在问题

你需要从一个网站获取和解析一个HTML文档，并查找其中的相关数据。你可以使用下面解决方法：

## 解决方法

使用 [Jsoup.connect(String url)](http://jsoup.org/apidocs/org/jsoup/Jsoup.html" \l "connect(java.lang.String)" \o "Creates a new Connection to a URL.)方法:

Document doc = Jsoup.connect("http://example.com/").get();

String title = doc.title();

## 说明

[connect(String url)](http://jsoup.org/apidocs/org/jsoup/Jsoup.html" \l "connect(java.lang.String)" \o "Creates a new Connection to a URL.) 方法创建一个新的 [Connection](http://jsoup.org/apidocs/org/jsoup/Connection.html" \o "A Connection provides a convenient interface to fetch content from the web, and parse them into Documents.), 和 [get()](http://jsoup.org/apidocs/org/jsoup/helper/HttpConnection.html" \l "get()) 取得和解析一个HTML文件。如果从该URL获取HTML时发生错误，便会抛出 IOException，应适当处理。

[Connection](http://jsoup.org/apidocs/org/jsoup/Connection.html" \o "A Connection provides a convenient interface to fetch content from the web, and parse them into Documents.) 接口还提供一个方法链来解决特殊请求，具体如下：

Document doc = Jsoup.connect("http://example.com")

.data("query", "Java")

.userAgent("Mozilla")

.cookie("auth", "token")

.timeout(3000)

.post();

这个方法只支持Web URLs (http和https 协议); 假如你需要从一个文件加载，可以使用 [parse(File in, String charsetName)](http://jsoup.org/apidocs/org/jsoup/Jsoup.html" \l "parse(java.io.File, java.lang.String)" \o "Parse the contents of a file as HTML.) 代替。

# 5.从一个文件加载一个文档

## 问题

在本机硬盘上有一个HTML文件，需要对它进行解析从中抽取数据或进行修改。

## 办法

可以使用静态 [Jsoup.parse(File in, String charsetName, String baseUri)](http://jsoup.org/apidocs/org/jsoup/Jsoup.html" \l "parse(java.io.File, java.lang.String, java.lang.String)" \o "Parse the contents of a file as HTML.) 方法：

File input = new File("/tmp/input.html");

Document doc = Jsoup.parse(input, "UTF-8", "http://example.com/");

## 说明

[parse(File in, String charsetName, String baseUri)](http://jsoup.org/apidocs/org/jsoup/Jsoup.html" \l "parse(java.io.File, java.lang.String, java.lang.String)" \o "Parse the contents of a file as HTML.) 这个方法用来加载和解析一个HTML文件。如在加载文件的时候发生错误，将抛出IOException，应作适当处理。

baseUri 参数用于解决文件中URLs是相对路径的问题。如果不需要可以传入一个空的字符串。

另外还有一个方法[parse(File in, String charsetName)](http://jsoup.org/apidocs/org/jsoup/Jsoup.html" \l "parse(java.io.File, java.lang.String)" \o "Parse the contents of a file as HTML.) ，它使用文件的路径做为 baseUri。 这个方法适用于如果被解析文件位于网站的本地文件系统，且相关链接也指向该文件系统。

# 6.使用DOM方法来遍历一个文档

## 问题

你有一个HTML文档要从中提取数据，并了解这个HTML文档的结构。

## 方法

将HTML解析成一个[Document](http://jsoup.org/apidocs/org/jsoup/nodes/Document.html" \o "A HTML Document.)之后，就可以使用类似于DOM的方法进行操作。示例代码：

File input = new File("/tmp/input.html");

Document doc = Jsoup.parse(input, "UTF-8", "http://example.com/");

Element content = doc.getElementById("content");

Elements links = content.getElementsByTag("a");

for (Element link : links) {

String linkHref = link.attr("href");

String linkText = link.text();

}

## 说明

Elements这个对象提供了一系列类似于DOM的方法来查找元素，抽取并处理其中的数据。具体如下：

### 查找元素

* [getElementById(String id)](http://jsoup.org/apidocs/org/jsoup/nodes/Element.html" \l "getElementById(java.lang.String)" \o "Find an element by ID, including or under this element.)
* [getElementsByTag(String tag)](http://jsoup.org/apidocs/org/jsoup/nodes/Element.html" \l "getElementsByTag(java.lang.String)" \o "Finds elements, including and recursively under this element, with the specified tag name.)
* [getElementsByClass(String className)](http://jsoup.org/apidocs/org/jsoup/nodes/Element.html" \l "getElementsByClass(java.lang.String)" \o "Find elements that have this class, including or under this element.)
* [getElementsByAttribute(String key)](http://jsoup.org/apidocs/org/jsoup/nodes/Element.html" \l "getElementsByAttribute(java.lang.String)" \o "Find elements that have a named attribute set.) (and related methods)
* Element siblings: [siblingElements()](http://jsoup.org/apidocs/org/jsoup/nodes/Element.html" \l "siblingElements()" \o "Get sibling elements.), [firstElementSibling()](http://jsoup.org/apidocs/org/jsoup/nodes/Element.html" \l "firstElementSibling()" \o "Gets the first element sibling of this element.), [lastElementSibling()](http://jsoup.org/apidocs/org/jsoup/nodes/Element.html" \l "lastElementSibling()" \o "Gets the last element sibling of this element); [nextElementSibling()](http://jsoup.org/apidocs/org/jsoup/nodes/Element.html" \l "nextElementSibling()" \o "Gets the next sibling element of this element.), [previousElementSibling()](http://jsoup.org/apidocs/org/jsoup/nodes/Element.html" \l "previousElementSibling()" \o "Gets the previous element sibling of this element.)
* Graph: [parent()](http://jsoup.org/apidocs/org/jsoup/nodes/Node.html" \l "parent()" \o "Gets this node's parent node.), [children()](http://jsoup.org/apidocs/org/jsoup/nodes/Element.html" \l "children()" \o "Get this element's child elements.), [child(int index)](http://jsoup.org/apidocs/org/jsoup/nodes/Element.html" \l "child(int)" \o "Get a child element of this element, by its 0-based index number.)

### 元素数据

* [attr(String key)](http://jsoup.org/apidocs/org/jsoup/select/Elements.html" \l "attr(java.lang.String)" \o "Get an attribute value from the first matched element that has the attribute.)获取属性[attr(String key, String value)](http://jsoup.org/apidocs/org/jsoup/select/Elements.html" \l "attr(java.lang.String, java.lang.String)" \o "Set an attribute on all matched elements.)设置属性
* [attributes()](http://jsoup.org/apidocs/org/jsoup/nodes/TextNode.html" \l "attributes())获取所有属性
* [id()](http://jsoup.org/apidocs/org/jsoup/nodes/Element.html" \l "id()" \o "Get the id attribute of this element.), [className()](http://jsoup.org/apidocs/org/jsoup/nodes/Element.html" \l "className()" \o "Gets the literal value of this element's "class" attribute, which may include multiple class names, space separated.) and [classNames()](http://jsoup.org/apidocs/org/jsoup/nodes/Element.html" \l "classNames()" \o "Get all of the element's class names.)
* [text()](http://jsoup.org/apidocs/org/jsoup/select/Elements.html" \l "text()" \o "Get the combined text of all the matched elements.)获取文本内容[text(String value)](http://jsoup.org/apidocs/org/jsoup/nodes/TextNode.html" \l "text(java.lang.String)" \o "Set the text content of this text node.) 设置文本内容
* [html()](http://jsoup.org/apidocs/org/jsoup/select/Elements.html" \l "html()" \o "Get the combined inner HTML of all matched elements.)获取元素内HTML[html(String value)](http://jsoup.org/apidocs/org/jsoup/select/Elements.html" \l "html(java.lang.String)" \o "Set the inner HTML of each matched element.)设置元素内的HTML内容
* [outerHtml()](http://jsoup.org/apidocs/org/jsoup/select/Elements.html" \l "outerHtml()" \o "Get the combined outer HTML of all matched elements.)获取元素外HTML内容
* [data()](http://jsoup.org/apidocs/org/jsoup/nodes/Element.html" \l "data()" \o "Get the combined data of this element.)获取数据内容（例如：script和style标签)
* [tag()](http://jsoup.org/apidocs/org/jsoup/nodes/Element.html" \l "tag()" \o "Get the Tag for this element.) and [tagName()](http://jsoup.org/apidocs/org/jsoup/nodes/Element.html" \l "tagName()" \o "Get the name of the tag for this element.)

### 操作HTML和文本

* [append(String html)](http://jsoup.org/apidocs/org/jsoup/select/Elements.html" \l "append(java.lang.String)" \o "Add the supplied HTML to the end of each matched element's inner HTML.), [prepend(String html)](http://jsoup.org/apidocs/org/jsoup/select/Elements.html" \l "prepend(java.lang.String)" \o "Add the supplied HTML to the start of each matched element's inner HTML.)
* [appendText(String text)](http://jsoup.org/apidocs/org/jsoup/nodes/Element.html" \l "appendText(java.lang.String)" \o "Create and append a new TextNode to this element.), [prependText(String text)](http://jsoup.org/apidocs/org/jsoup/nodes/Element.html" \l "prependText(java.lang.String)" \o "Create and prepend a new TextNode to this element.)
* [appendElement(String tagName)](http://jsoup.org/apidocs/org/jsoup/nodes/Element.html" \l "appendElement(java.lang.String)" \o "Create a new element by tag name, and add it as the last child.), [prependElement(String tagName)](http://jsoup.org/apidocs/org/jsoup/nodes/Element.html" \l "prependElement(java.lang.String)" \o "Create a new element by tag name, and add it as the first child.)
* [html(String value)](http://jsoup.org/apidocs/org/jsoup/select/Elements.html" \l "html(java.lang.String)" \o "Set the inner HTML of each matched element.)

# 7.使用选择器语法来查找元素

## 问题

你想使用类似于CSS或jQuery的语法来查找和操作元素。

## 方法

可以使用[Element.select(String selector)](http://jsoup.org/apidocs/org/jsoup/nodes/Element.html" \l "select(java.lang.String)" \o "Find elements that match the Selector query, with this element as the starting context.) 和 [Elements.select(String selector)](http://jsoup.org/apidocs/org/jsoup/select/Elements.html" \l "select(java.lang.String)" \o "Find matching elements within this element list.) 方法实现：

File input = new File("/tmp/input.html");

Document doc = Jsoup.parse(input, "UTF-8", "http://example.com/");

Elements links = doc.select("a[href]"); //带有href属性的a元素

Elements pngs = doc.select("img[src$=.png]");

//扩展名为.png的图片

Element masthead = doc.select("div.masthead").first();

//class等于masthead的div标签

Elements resultLinks = doc.select("h3.r > a"); //在h3元素之后的a元素

## 说明

jsoup elements对象支持类似于[CSS](http://www.w3.org/TR/2009/PR-css3-selectors-20091215/) (或[jquery](http://jquery.com/))的选择器语法，来实现非常强大和灵活的查找功能。.

这个select 方法在[Document](http://jsoup.org/apidocs/org/jsoup/nodes/Document.html" \o "A HTML Document.), [Element](http://jsoup.org/apidocs/org/jsoup/nodes/Element.html" \o "A HTML element consists of a tag name, attributes, and child nodes (including text nodes and other elements).),或[Elements](http://jsoup.org/apidocs/org/jsoup/select/Elements.html" \o "A list of Elements, with methods that act on every element in the list.)对象中都可以使用。且是上下文相关的，因此可实现指定元素的过滤，或者链式选择访问。

Select方法将返回一个[Elements](http://jsoup.org/apidocs/org/jsoup/select/Elements.html" \o "A list of Elements, with methods that act on every element in the list.)集合，并提供一组方法来抽取和处理结果。

### Selector选择器概述

* tagname: 通过标签查找元素，比如：a
* ns|tag: 通过标签在命名空间查找元素，比如：可以用 fb|name 语法来查找 <fb:name> 元素
* #id: 通过ID查找元素，比如：#logo
* .class: 通过class名称查找元素，比如：.masthead
* [attribute]: 利用属性查找元素，比如：[href]
* [^attr]: 利用属性名前缀来查找元素，比如：可以用[^data-] 来查找带有HTML5 Dataset属性的元素
* [attr=value]: 利用属性值来查找元素，比如：[width=500]
* [attr^=value], [attr$=value], [attr\*=value]: 利用匹配属性值开头、结尾或包含属性值来查找元素，比如：[href\*=/path/]
* [attr~=regex]: 利用属性值匹配正则表达式来查找元素，比如： img[src~=(?i)\.(png|jpe?g)]
* \*: 这个符号将匹配所有元素

### Selector选择器组合使用

* el#id: 元素+ID，比如： div#logo
* el.class: 元素+class，比如： div.masthead
* el[attr]: 元素+class，比如： a[href]
* 任意组合，比如：a[href].highlight
* ancestor child: 查找某个元素下子元素，比如：可以用.body p 查找在"body"元素下的所有 p元素
* parent > child: 查找某个父元素下的直接子元素，比如：可以用div.content > p 查找 p 元素，也可以用body > \* 查找body标签下所有直接子元素
* siblingA + siblingB: 查找在A元素之前第一个同级元素B，比如：div.head + div
* siblingA ~ siblingX: 查找A元素之前的同级X元素，比如：h1 ~ p
* el, el, el:多个选择器组合，查找匹配任一选择器的唯一元素，例如：div.masthead, div.logo

### 伪选择器selectors

* :lt(n): 查找哪些元素的同级索引值（它的位置在DOM树中是相对于它的父节点）小于n，比如：td:lt(3) 表示小于三列的元素
* :gt(n):查找哪些元素的同级索引值大于n，比如： div p:gt(2)表示哪些div中有包含2个以上的p元素
* :eq(n): 查找哪些元素的同级索引值与n相等，比如：form input:eq(1)表示包含一个input标签的Form元素
* :has(seletor): 查找匹配选择器包含元素的元素，比如：div:has(p)表示哪些div包含了p元素
* :not(selector): 查找与选择器不匹配的元素，比如： div:not(.logo) 表示不包含 class=logo 元素的所有 div 列表
* **:contains(text): 查找包含给定文本的元素，搜索不区分大不写，比如： p:contains(jsoup)**
* **:containsOwn(text): 查找直接包含给定文本的元素**
* :matches(regex): 查找哪些元素的文本匹配指定的正则表达式，比如：div:matches((?i)login)
* :matchesOwn(regex): 查找自身包含文本匹配指定正则表达式的元素
* 注意：上述伪选择器索引是从0开始的，也就是说第一个元素索引值为0，第二个元素index为1等

可以查看[Selector](http://jsoup.org/apidocs/org/jsoup/select/Selector.html" \o "CSS-like element selector, that finds elements matching a query.) API参考来了解更详细的内容

# 8.从元素抽取属性，文本和HTML

## 问题

在解析获得一个Document实例对象，并查找到一些元素之后，你希望取得在这些元素中的数据。

## 方法

* 要取得一个属性的值，可以使用[Node.attr(String key)](http://jsoup.org/apidocs/org/jsoup/nodes/Node.html" \l "attr(java.lang.String)" \o "Get an attribute's value by its key.) 方法
* 对于一个元素中的文本，可以使用[Element.text()](http://jsoup.org/apidocs/org/jsoup/nodes/Element.html" \l "text()" \o "Gets the combined text of this element and all its children.)方法
* 对于要取得元素或属性中的HTML内容，可以使用[Element.html()](http://jsoup.org/apidocs/org/jsoup/nodes/Element.html" \l "html()" \o "Retrieves the element's inner HTML.), 或 [Node.outerHtml()](http://jsoup.org/apidocs/org/jsoup/nodes/Node.html" \l "outerHtml()" \o "Get the outer HTML of this node.)方法

示例：

String html = "<p>An <a href='http://example.com/'><b>example</b></a> link.</p>";

Document doc = Jsoup.parse(html);//解析HTML字符串返回一个Document实现

Element link = doc.select("a").first();//查找第一个a元素

String text = doc.body().text(); // "An example link"//取得字符串中的文本

String linkHref = link.attr("href"); // "http://example.com/"//取得链接地址

String linkText = link.text(); // "example""//取得链接地址中的文本

String linkOuterH = link.outerHtml();

// "<a href="http://example.com"><b>example</b></a>"

String linkInnerH = link.html(); // "<b>example</b>"//取得链接内的html内容

## 说明

上述方法是元素数据访问的核心办法。此外还其它一些方法可以使用：

* [Element.id()](http://jsoup.org/apidocs/org/jsoup/nodes/Element.html" \l "id()" \o "Get the id attribute of this element.)
* [Element.tagName()](http://jsoup.org/apidocs/org/jsoup/nodes/Element.html" \l "tagName()" \o "Get the name of the tag for this element.)
* [Element.className()](http://jsoup.org/apidocs/org/jsoup/nodes/Element.html" \l "className()" \o "Gets the literal value of this element's "class" attribute, which may include multiple class names, space separated.) and [Element.hasClass(String className)](http://jsoup.org/apidocs/org/jsoup/nodes/Element.html" \l "hasClass(java.lang.String)" \o "Tests if this element has a class.)

这些访问器方法都有相应的setter方法来更改数据.

## 参见

* [Element](http://jsoup.org/apidocs/org/jsoup/nodes/Element.html" \o "A HTML element consists of a tag name, attributes, and child nodes (including text nodes and other elements).)和[Elements](http://jsoup.org/apidocs/org/jsoup/select/Elements.html" \o "A list of Elements, with methods that act on every element in the list.)集合类的参考文档
* [URLs处理](http://www.open-open.com/jsoup/working-with-urls.htm)
* [使用CSS选择器语法来查找元素](http://www.open-open.com/jsoup/selector-syntax.htm)

# 9.处理URLs

## 问题

你有一个包含相对URLs路径的HTML文档，需要将这些相对路径转换成绝对路径的URLs。

## 方法

1. 在你解析文档时确保有指定base URI，然后
2. 使用 abs: 属性前缀来取得包含base URI的绝对路径。代码如下：

Document doc = Jsoup.connect("http://www.open-open.com").get();

Element link = doc.select("a").first();

String relHref = link.attr("href"); // == "/"

String absHref = link.attr("abs:href"); // "http://www.open-open.com/"

## 说明

在HTML元素中，URLs经常写成相对于文档位置的相对路径： <a href="/download">...</a>. 当你使用 [Node.attr(String key)](http://jsoup.org/apidocs/org/jsoup/nodes/Node.html" \l "attr(java.lang.String)" \o "Get an attribute's value by its key.) 方法来取得a元素的href属性时，它将直接返回在HTML源码中指定定的值。

假如你需要取得一个绝对路径，需要在属性名前加 abs: 前缀。这样就可以返回包含根路径的URL地址attr("abs:href")

因此，在解析HTML文档时，定义base URI非常重要。

如果你不想使用abs: 前缀，还有一个方法能够实现同样的功能 [Node.absUrl(String key)](http://jsoup.org/apidocs/org/jsoup/nodes/Node.html" \l "absUrl(java.lang.String)" \o "Get an absolute URL from a URL attribute that may be relative (i.e.)。

# 10.示例程序: 获取所有链接

这个示例程序将展示如何从一个URL获得一个页面。然后提取页面中的所有链接、图片和其它辅助内容。并检查URLs和文本信息。

运行下面程序需要指定一个URLs作为参数

package org.jsoup.examples;

import org.jsoup.Jsoup;

import org.jsoup.helper.Validate;

import org.jsoup.nodes.Document;

import org.jsoup.nodes.Element;

import org.jsoup.select.Elements;

import java.io.IOException;

/\*\*

\* Example program to list links from a URL.

\*/

public class ListLinks {

public static void main(String[] args) throws IOException {

Validate.isTrue(args.length == 1, "usage: supply url to fetch");

String url = args[0];

print("Fetching %s...", url);

Document doc = Jsoup.connect(url).get();

Elements links = doc.select("a[href]");

Elements media = doc.select("[src]");

Elements imports = doc.select("link[href]");

print("\nMedia: (%d)", media.size());

for (Element src : media) {

if (src.tagName().equals("img"))

print(" \* %s: <%s> %sx%s (%s)",

src.tagName(), src.attr("abs:src"), src.attr("width"), src.attr("height"),

trim(src.attr("alt"), 20));

else

print(" \* %s: <%s>", src.tagName(), src.attr("abs:src"));

}

print("\nImports: (%d)", imports.size());

for (Element link : imports) {

print(" \* %s <%s> (%s)", link.tagName(),link.attr("abs:href"), link.attr("rel"));

}

print("\nLinks: (%d)", links.size());

for (Element link : links) {

print(" \* a: <%s> (%s)", link.attr("abs:href"), trim(link.text(), 35));

}

}

private static void print(String msg, Object... args) {

System.out.println(String.format(msg, args));

}

private static String trim(String s, int width) {

if (s.length() > width)

return s.substring(0, width-1) + ".";

else

return s;

}

}

[org/jsoup/examples/ListLinks.java](http://github.com/jhy/jsoup/blob/master/src/main/java/org/jsoup/examples/ListLinks.java)

## 示例输入结果

Fetching http://news.ycombinator.com/...

Media: (38)

\* img: <http://ycombinator.com/images/y18.gif> 18x18 ()

\* img: <http://ycombinator.com/images/s.gif> 10x1 ()

\* img: <http://ycombinator.com/images/grayarrow.gif> x ()

\* img: <http://ycombinator.com/images/s.gif> 0x10 ()

\* script: <http://www.co2stats.com/propres.php?s=1138>

\* img: <http://ycombinator.com/images/s.gif> 15x1 ()

\* img: <http://ycombinator.com/images/hnsearch.png> x ()

\* img: <http://ycombinator.com/images/s.gif> 25x1 ()

\* img: <http://mixpanel.com/site\_media/images/mixpanel\_partner\_logo\_borderless.gif> x (Analytics by Mixpan.)

Imports: (2)

\* link <http://ycombinator.com/news.css> (stylesheet)

\* link <http://ycombinator.com/favicon.ico> (shortcut icon)

Links: (141)

\* a: <http://ycombinator.com> ()

\* a: <http://news.ycombinator.com/news> (Hacker News)

\* a: <http://news.ycombinator.com/newest> (new)

\* a: <http://news.ycombinator.com/newcomments> (comments)

\* a: <http://news.ycombinator.com/leaders> (leaders)

\* a: <http://news.ycombinator.com/jobs> (jobs)

\* a: <http://news.ycombinator.com/submit> (submit)

\* a: <http://news.ycombinator.com/x?fnid=JKhQjfU7gW> (login)

\* a: <http://news.ycombinator.com/vote?for=1094578&dir=up&whence=%6e%65%77%73> ()

\* a: <http://www.readwriteweb.com/archives/facebook\_gets\_faster\_debuts\_homegrown\_php\_compiler.php?utm\_source=feedburner&utm\_medium=feed&utm\_campaign=Feed%3A+readwriteweb+%28ReadWriteWeb%29&utm\_content=Twitter> (Facebook speeds up PHP)

\* a: <http://news.ycombinator.com/user?id=mcxx> (mcxx)

\* a: <http://news.ycombinator.com/item?id=1094578> (9 comments)

\* a: <http://news.ycombinator.com/vote?for=1094649&dir=up&whence=%6e%65%77%73> ()

\* a: <http://groups.google.com/group/django-developers/msg/a65fbbc8effcd914> ("Tough. Django produces XHTML.")

\* a: <http://news.ycombinator.com/user?id=andybak> (andybak)

\* a: <http://news.ycombinator.com/item?id=1094649> (3 comments)

\* a: <http://news.ycombinator.com/vote?for=1093927&dir=up&whence=%6e%65%77%73> ()

\* a: <http://news.ycombinator.com/x?fnid=p2sdPLE7Ce> (More)

\* a: <http://news.ycombinator.com/lists> (Lists)

\* a: <http://news.ycombinator.com/rss> (RSS)

\* a: <http://ycombinator.com/bookmarklet.html> (Bookmarklet)

\* a: <http://ycombinator.com/newsguidelines.html> (Guidelines)

\* a: <http://ycombinator.com/newsfaq.html> (FAQ)

\* a: <http://ycombinator.com/newsnews.html> (News News)

\* a: <http://news.ycombinator.com/item?id=363> (Feature Requests)

\* a: <http://ycombinator.com> (Y Combinator)

\* a: <http://ycombinator.com/w2010.html> (Apply)

\* a: <http://ycombinator.com/lib.html> (Library)

\* a: <http://www.webmynd.com/html/hackernews.html> ()

\* a: <http://mixpanel.com/?from=yc> ()

# 11.设置属性的值

## 问题

在你解析一个Document之后可能想修改其中的某些属性值，然后再保存到磁盘或都输出到前台页面。

## 方法

可以使用属性设置方法 [Element.attr(String key, String value)](http://jsoup.org/apidocs/org/jsoup/nodes/Element.html" \l "attr(java.lang.String, java.lang.String)" \o "Set an attribute value on this element.), 和 [Elements.attr(String key, String value)](http://jsoup.org/apidocs/org/jsoup/select/Elements.html" \l "attr(java.lang.String, java.lang.String)" \o "Set an attribute on all matched elements.).

假如你需要修改一个元素的 class 属性，可以使用 [Element.addClass(String className)](http://jsoup.org/apidocs/org/jsoup/nodes/Element.html" \l "addClass(java.lang.String)" \o "Add a class name to this element's class attribute.) 和 [Element.removeClass(String className)](http://jsoup.org/apidocs/org/jsoup/nodes/Element.html" \l "removeClass(java.lang.String)" \o "Remove a class name from this element's class attribute.) 方法。

[Elements](http://jsoup.org/apidocs/org/jsoup/select/Elements.html" \o "A list of Elements, with methods that act on every element in the list.) 提供了批量操作元素属性和class的方法，比如：要为div中的每一个a元素都添加一个 rel="nofollow" 可以使用如下方法：

doc.select("div.comments a").attr("rel", "nofollow");

## 说明

与[Element](http://jsoup.org/apidocs/org/jsoup/nodes/Element.html" \o "A HTML element consists of a tag name, attributes, and child nodes (including text nodes and other elements).)中的其它方法一样，attr 方法也是返回当 [Element](http://jsoup.org/apidocs/org/jsoup/nodes/Element.html" \o "A HTML element consists of a tag name, attributes, and child nodes (including text nodes and other elements).) (或在使用选择器是返回 [Elements](http://jsoup.org/apidocs/org/jsoup/select/Elements.html" \o "A list of Elements, with methods that act on every element in the list.) 集合)。这样能够很方便使用方法连用的书写方式。比如：

doc.select("div.masthead").attr("title", "jsoup").addClass("round-box");

# 12.设置一个元素的HTML内容

## 问题

你需要一个元素中的HTML内容

## 方法

可以使用[Element](http://jsoup.org/apidocs/org/jsoup/nodes/Element.html" \o "A HTML element consists of a tag name, attributes, and child nodes (including text nodes and other elements).)中的HTML设置方法具体如下：

Element div = doc.select("div").first(); // <div></div>

div.html("<p>lorem ipsum</p>"); // <div><p>lorem ipsum</p></div>

div.prepend("<p>First</p>");//在div前添加html内容

div.append("<p>Last</p>");//在div之后添加html内容

// 添完后的结果: <div><p>First</p><p>lorem ipsum</p><p>Last</p></div>

Element span = doc.select("span").first(); // <span>One</span>

span.wrap("<li><a href='http://example.com/'></a></li>");

// 添完后的结果: <li><a href="http://example.com"><span>One</span></a></li>

## 说明

* [Element.html(String html)](http://jsoup.org/apidocs/org/jsoup/nodes/Element.html" \l "html(java.lang.String)" \o "Set this element's inner HTML.) 这个方法将先清除元素中的HTML内容，然后用传入的HTML代替。
* [Element.prepend(String first)](http://jsoup.org/apidocs/org/jsoup/nodes/Element.html" \l "prepend(java.lang.String)" \o "Add inner HTML into this element.) 和 [Element.append(String last)](http://jsoup.org/apidocs/org/jsoup/nodes/Element.html" \l "append(java.lang.String)" \o "Add inner HTML to this element.) 方法用于在分别在元素内部HTML的前面和后面添加HTML内容
* [Element.wrap(String around)](http://jsoup.org/apidocs/org/jsoup/nodes/Element.html" \l "wrap(java.lang.String)" \o "Wrap the supplied HTML around this element.) 对元素包裹一个外部HTML内容。

## 参见

可以查看API参考文档中 [Element.prependElement(String tag)](http://jsoup.org/apidocs/org/jsoup/nodes/Element.html" \l "prependElement(java.lang.String)" \o "Create a new element by tag name, and add it as the first child.)和[Element.appendElement(String tag)](http://jsoup.org/apidocs/org/jsoup/nodes/Element.html" \l "appendElement(java.lang.String)" \o "Create a new element by tag name, and add it as the last child.) 方法来创建新的元素并作为文档的子元素插入其中。

# 13.设置元素的文本内容

## 问题

你需要修改一个HTML文档中的文本内容

## 方法

可以使用[Element](http://jsoup.org/apidocs/org/jsoup/nodes/Element.html" \o "A HTML element consists of a tag name, attributes, and child nodes (including text nodes and other elements).)的设置方法：:

Element div = doc.select("div").first(); // <div></div>

div.text("five > four"); // <div>five &gt; four</div>

div.prepend("First ");

div.append(" Last");

// now: <div>First five &gt; four Last</div>

## 说明

文本设置方法与 [HTML setter](http://jsoup.org/cookbook/modifying-data/set-html) 方法一样：

* [Element.text(String text)](http://jsoup.org/apidocs/org/jsoup/nodes/Element.html" \l "text(java.lang.String)" \o "Set the text of this element.) 将清除一个元素中的内部HTML内容，然后提供的文本进行代替
* [Element.prepend(String first)](http://jsoup.org/apidocs/org/jsoup/nodes/Element.html" \l "prepend(java.lang.String)" \o "Add inner HTML into this element.) 和 [Element.append(String last)](http://jsoup.org/apidocs/org/jsoup/nodes/Element.html" \l "append(java.lang.String)" \o "Add inner HTML to this element.) 将分别在元素的内部html前后添加文本节点。

对于传入的文本如果含有像 <, > 等这样的字符，将以文本处理，而非HTML。

# 14.消除不受信任的HTML (来防止XSS攻击)

## 问题

在做网站的时候，经常会提供用户评论的功能。有些不怀好意的用户，会搞一些脚本到评论内容中，而这些脚本可能会破坏整个页面的行为，更严重的是获取一些机要信息，此时需要清理该HTML，以避免跨站脚本[cross-site scripting](http://en.wikipedia.org/wiki/Cross-site_scripting)攻击（XSS）。

## 方法

使用jsoup HTML [Cleaner](http://jsoup.org/apidocs/org/jsoup/safety/Cleaner.html" \o "The whitelist based HTML cleaner.) 方法进行清除，但需要指定一个可配置的 [Whitelist](http://jsoup.org/apidocs/org/jsoup/safety/Whitelist.html" \o "Whitelists define what HTML (elements and attributes) to allow through the cleaner.)。

String unsafe =

"<p><a href='http://example.com/' onclick='stealCookies()'>Link</a></p>";

String safe = Jsoup.clean(unsafe, Whitelist.basic());

// now: <p><a href="http://example.com/" rel="nofollow">Link</a></p>

## 说明

XSS又叫CSS (Cross Site Script) ，跨站脚本攻击。它指的是恶意攻击者往Web页面里插入恶意html代码，当用户浏览该页之时，嵌入其中Web里面的html代码会被执行，从而达到恶意攻击用户的特殊目的。XSS属于被动式的攻击，因为其被动且不好利用，所以许多人常忽略其危害性。所以我们经常只让用户输入纯文本的内容，但这样用户体验就比较差了。

一个更好的解决方法就是使用一个富文本编辑器WYSIWYG如[CKEditor](http://ckeditor.com/) 和 [TinyMCE](http://tinymce.moxiecode.com/)。这些可以输出HTML并能够让用户可视化编辑。虽然他们可以在客户端进行校验，但是这样还不够安全，需要在服务器端进行校验并清除有害的HTML代码，这样才能确保输入到你网站的HTML是安全的。否则，攻击者能够绕过客户端的Javascript验证，并注入不安全的HMTL直接进入您的网站。

jsoup的whitelist清理器能够在服务器端对用户输入的HTML进行过滤，只输出一些安全的标签和属性。

jsoup提供了一系列的[Whitelist](http://jsoup.org/apidocs/org/jsoup/safety/Whitelist.html" \o "Whitelists define what HTML (elements and attributes) to allow through the cleaner.)基本配置，能够满足大多数要求；但如有必要，也可以进行修改，不过要小心。

这个cleaner非常好用不仅可以避免XSS攻击，还可以限制用户可以输入的标签范围。

## 参见

* 参阅[XSS cheat sheet](http://ha.ckers.org/xss.html) ，有一个例子可以了解为什么不能使用正则表达式，而采用安全的whitelist parser-based清理器才是正确的选择。
* 参阅[Cleaner](http://jsoup.org/apidocs/org/jsoup/safety/Cleaner.html" \o "The whitelist based HTML cleaner.) ，了解如何返回一个 [Document](http://jsoup.org/apidocs/org/jsoup/nodes/Document.html" \o "A HTML Document.) 对象，而不是字符串
* 参阅[Whitelist](http://jsoup.org/apidocs/org/jsoup/safety/Whitelist.html" \o "Whitelists define what HTML (elements and attributes) to allow through the cleaner.)，了解如何创建一个自定义的whitelist
* [nofollow](http://en.wikipedia.org/wiki/Nofollow) 链接属性了解