jsoup - 简介介绍

jsoup是一个用于解析、提取、操作HTML的开源Java函式库，它提供了一个非常方便的API，可以使用DOM，CSS和类似jquery的方法来提取和处理数据。它实现了HTML5规范，并将HTML解析为与浏览器相同的DOM，本参考将带您了解jsoup库中提供的简单实用的方法。

本参考资料已为初学者准备，以帮助他们了解与jsoup库中可用功能相关的基本功能。

在开始使用本参考中给出的各种类型的示例进行练习之前，我假设您已经了解基本的Java编程。

下一篇:jsoup - 环境设置

jsoup - 环境设置介绍

步骤1:验证机器中的Java安装

首先，打开控制台并根据您正在使用的操作系统执行Java命令。

OS	Task	Command
Windows	Open Command Console	c:\> java -version
Linux	Open Command Terminal	$java -version
Mac	Open Terminal	machine:

让我们验证所有操作系统的输出-

OS	输出
Windows	java版本" 1.6.0_21" Java(TM)SE运行时环境(内部版本1.6.0_21-b07)
Linux	java版本" 1.6.0_21" Java(TM)SE运行时环境(内部版本1.6.0_21-b07)
Mac	java版本" 1.6.0_21" Java(TM)SE运行时环境(内部版本1.6.0_21-b07)

如果您的系统上未安装Java,请从以下链接下载Java软件开发工具包(SDK): https://www.oracle.com 。我们假定Java 1.6.0_21作为本教程的安装版本。

步骤2:设定JAVA环境

将 JAVA_HOME 环境变量设置为指向计算机上Java安装位置的基本目录位置。例如。

OS	输出
Windows	将环境变量JAVA_HOME设置为 C:\Program Files\Java\jdk1.6.0_21
Linux	export JAVA_HOME =/usr/local/java-current
Mac	export JAVA_HOME =/Library/Java/Home

将Java编译器位置附加到系统路径。

OS	输出
Windows	在系统变量 Path 的末尾附加字符串 C:\Program Files\Java\jdk1.6.0_21\bin
Linux	export PATH=$PATH:$JAVA_HOME/bin/
Mac	不需要

如上所述，使用命令 java -version 验证Java安装。

步骤3:下载jsoup库

从Maven存储库下载最新版本的jsoup jar文件，在编写本教程时，我们已经下载了jsoup-1.8.3.jar并将其复制到C:\> jsoup文件夹中。

步骤4:设置jsoup环境

将 JSOUP_HOME 环境变量设置为指向jsoup jar在您的计算机上存储的基本目录位置，假设我们已经将jsoup-1.8.3.jar存储在JSOUP文件夹中。

Sr.No	OS & 描述
1	Windows 将环境变量JSOUP_HOME设置为 C:\JSOUP
2	Linux export JSOUP_HOME =/usr/local/JSOUP
3	Mac export JSOUP_HOME =/Library/JSOUP

步骤5:设置CLASSPATH变量

将 CLASSPATH 环境变量设置为指向JSOUP jar位置。

Sr.No	OS & 描述
1	Windows 将环境变量CLASSPATH设置为％CLASSPATH％;％JSOUP_HOME％\jsoup-1.8.3.jar;。;
2	Linux export CLASSPATH=$CLASSPATH:$JSOUP_HOME/jsoup-1.8.3.jar:。
3	Mac export CLASSPATH=$CLASSPATH:$JSOUP_HOME/jsoup-1.8.3.jar:。

上一篇:jsoup - 简介

jsoup - 解析字符串介绍

以下示例将HTML字符串解析为Document对象。

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

parse(String html)方法将输入的HTML解析为新的文档。该文档对象可用于遍历并获取html dom的详细信息。

Jsoup.parse 示例

使用您选择的任何编辑器在C:/> jsoup中创建以下Java程序。

JsoupTester.java

```
import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;
import org.jsoup.nodes.Element;
import org.jsoup.select.Elements;

public class JsoupTester {
   public static void main(String[] args) {

      String html = "<html><head><title>Sample Title</title></head>"
         + "<body><p>Sample Content</p></body></html>";
      Document document = Jsoup.parse(html);
      System.out.println(document.title());
      Elements paragraphs = document.getElementsByTag("p");
      for (Element paragraph : paragraphs) {
            System.out.println(paragraph.text());
      }
   }
}
```

使用 javac 编译器编译类，如下所示:

```
C:\jsoup>javac JsoupTester.java
```

现在运行JsoupTester以查看输出。

```
C:\jsoup>java JsoupTester
```

查看输出。

```
Sample Title
Sample Content
```

下一篇:jsoup - 解析正文

jsoup - 解析正文介绍

下面的示例将解析HTML为新的文档(Document)，然后获取html body的Element对象。

```
Document document=Jsoup.parseBodyFragment(html);
Element body=document.body();
```

parseBodyFragment(String html)方法将输入的HTML解析为新的文档，该文档对象可用于遍历并获取html正文片段的详细信息。

Document.body 示例

使用您选择的任何编辑器在C:/> jsoup中创建以下Java程序。

JsoupTester.java

```
import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;
import org.jsoup.nodes.Element;
import org.jsoup.select.Elements;

public class JsoupTester {
   public static void main(String[] args) {

      String html = "<div><p>Sample Content</p>";
      Document document = Jsoup.parseBodyFragment(html);
      Element body = document.body();
      Elements paragraphs = body.getElementsByTag("p");
      for (Element paragraph : paragraphs) {
         System.out.println(paragraph.text());
      }
   }
}
```

使用 javac 编译器编译类，如下所示:

```
C:\jsoup>javac JsoupTester.java
```

现在运行JsoupTester以查看输出。

```
C:\jsoup>java JsoupTester
```

查看输出。

```
Sample Content
```

上一篇:jsoup - 解析字符串

下一篇:jsoup - 加载URL

jsoup - 加载URL介绍

下面的示例将展示从URL链接中获取HTML内容，返回一个新的文档(Document)，然后查找其数据。

```
String url="http://www.google.com";
Document document=Jsoup.connect(url).get();
```

connect(url)方法创建与url的连接，get()方法返回所请求URL的html内容。

Jsoup.connect示例

使用您选择的任何编辑器在C:/> jsoup中创建以下Java程序。

JsoupTester.java

```
import java.io.IOException;

import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;

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

      String url = "http://www.google.com";
      Document document = Jsoup.connect(url).get();
      System.out.println(document.title());
   }
}
```

使用 javac 编译器编译类，如下所示:

```
C:\jsoup>javac JsoupTester.java
```

现在运行JsoupTester以查看输出。

```
C:\jsoup>java JsoupTester
```

查看输出。

```
Google
```

上一篇:jsoup - 解析正文

下一篇:jsoup - 加载文件

jsoup - 加载文件介绍

以下示例将从本地文件加载HTML文件，返回一个Document文档，然后查找其数据。

```
File input=new File(xxxxx);
Document document=Jsoup.parse(input, "UTF-8");
```

Jsoup.parse示例

使用您选择的任何编辑器在C:/> jsoup中创建以下Java程序。

JsoupTester.java

```
import java.io.File;
import java.io.IOException;
import java.net.URISyntaxException;
import java.net.URL;

import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;

public class JsoupTester {
   public static void main(String[] args) throws IOException, URISyntaxException {

      URL path = ClassLoader.getSystemResource("test.htm");
      File input = new File(path.toURI());
      Document document = Jsoup.parse(input, "UTF-8");
      System.out.println(document.title());
   }
}
```

在C:\jsoup文件夹中创建以下test.htm文件。

```
<html>
   <head>
      <title>Sample Title</title>
   </head>
   <body>
      <p>Sample Content</p>
   </body>
</html>
```

使用 javac 编译器编译类，如下所示:

```
C:\jsoup>javac JsoupTester.java
```

现在运行JsoupTester以查看输出。

```
C:\jsoup>java JsoupTester
```

查看输出。

```
Sample Title
```

上一篇:jsoup - 加载URL

jsoup - 使用DOM方法介绍

以下示例将HTML解析为Document对象之后，使用类似DOM的方法获取元素信息。

```
Document document=Jsoup.parse(html);
Element sampleDiv=document.getElementById("sampleDiv");
Elements links=sampleDiv.getElementsByTag("a");
```

parse(String html)方法将输入的HTML解析为新的文档，该文档对象可用于遍历并获取html dom的详细信息。

Document.getElementById示例

使用您选择的任何编辑器在C:/> jsoup中创建以下Java程序。

JsoupTester.java

```
import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;
import org.jsoup.nodes.Element;
import org.jsoup.select.Elements;

public class JsoupTester {
   public static void main(String[] args) {

      String html = "<html><head><title>Sample Title</title></head>"
         + "<body>"
         + "<p>Sample Content</p>"
         + "<div id='sampleDiv'><a href='www.google.com'>Google</a></div>"
         +"</body></html>";
      Document document = Jsoup.parse(html);
      System.out.println(document.title());
      Elements paragraphs = document.getElementsByTag("p");
      for (Element paragraph : paragraphs) {
         System.out.println(paragraph.text());
      }

      Element sampleDiv = document.getElementById("sampleDiv");
      System.out.println("Data: " + sampleDiv.text());
      Elements links = sampleDiv.getElementsByTag("a");

      for (Element link : links) {
         System.out.println("Href: " + link.attr("href"));
         System.out.println("Text: " + link.text());
      }
   }
}
```

使用 javac 编译器编译类，如下所示:

```
C:\jsoup>javac JsoupTester.java
```

现在运行JsoupTester以查看输出。

```
C:\jsoup>java JsoupTester
```

查看输出。

```
Sample Title
Sample Content
Data: Google
Href: www.google.com
Text: Google
```

下一篇:jsoup - 使用选择器语法

jsoup - 使用选择器语法介绍

以下示例将HTML解析为Document对象之后使用Selector方法操作元素， jsoup支持类似于CSS Selector选择器。

```
Document document=Jsoup.parse(html);
//a with href
Elements links=document.select("a[href]");
```

document.select(expression)方法解析给定的CSS Selector表达式，以选择html dom元素。

Document.select 示例

使用您选择的任何编辑器在C:/> jsoup中创建以下Java程序。

JsoupTester.java

```
import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;
import org.jsoup.nodes.Element;
import org.jsoup.select.Elements;

public class JsoupTester {
   public static void main(String[] args) {

      String html = "<html><head><title>Sample Title</title></head>"
         + "<body>"
         + "<p>Sample Content</p>"
         + "<div id='sampleDiv'><a href='www.google.com'>Google</a>"
         + "<h3><a>Sample</a><h3>"
         +"</div>"
         + "<div id='imageDiv' class='header'><img name='google' src='google.png?
imageView2/0/q/75|watermark/2/text/bGVhcm5may5jb20=/font/Y29uc29sYXM=/fontsize/400/fill/I0YxMTQxNA==/dissolve
/100/gravity/SouthEast/dx/10/dy/10' />"
         + "<img name='yahoo' src='yahoo.jpg?
imageView2/0/q/75|watermark/2/text/bGVhcm5may5jb20=/font/Y29uc29sYXM=/fontsize/400/fill/I0YxMTQxNA==/dissolve
/100/gravity/SouthEast/dx/10/dy/10' />"
         +"</div>"
         +"</body></html>";
      Document document = Jsoup.parse(html);

      //a with href
      Elements links = document.select("a[href]");

      for (Element link : links) {
         System.out.println("Href: " + link.attr("href"));
         System.out.println("Text: " + link.text());
      }

      //img with src ending .png?
imageView2/0/q/75|watermark/2/text/bGVhcm5may5jb20=/font/Y29uc29sYXM=/fontsize/400/fill/I0YxMTQxNA==/dissolve
/100/gravity/SouthEast/dx/10/dy/10
      Elements pngs = document.select("img[src$=.png?
imageView2/0/q/75|watermark/2/text/bGVhcm5may5jb20=/font/Y29uc29sYXM=/fontsize/400/fill/I0YxMTQxNA==/dissolve
/100/gravity/SouthEast/dx/10/dy/10]");

      for (Element png : pngs) {
         System.out.println("Name: " + png.attr("name"));
      }

      //div with class=header
      Element headerDiv = document.select("div.header").first();
      System.out.println("Id: " + headerDiv.id());

      //direct a after h3
      Elements sampleLinks = document.select("h3 > a");

      for (Element link : sampleLinks) {
```

```
      for (Element link : sampleLinks) {
         System.out.println("Text: " + link.text());
      }
   }
}
```

使用 javac 编译器编译类，如下所示:

```
C:\jsoup>javac JsoupTester.java
```

现在运行JsoupTester以查看输出。

```
C:\jsoup>java JsoupTester
```

查看输出。

```
Href: www.google.com
Text: Google
Name: google
Id: imageDiv
Text: Sample
```

上一篇:jsoup - 使用DOM方法　　下一篇:jsoup - 提取属性

jsoup - 提取属性介绍

以下示例将HTML解析为Document对象后，使用Elements方法来获取dom元素的属性。

```
Document document=Jsoup.parse(html);
Element link=document.select("a").first();
System.out.println("Href: " + link.attr("href"));
```

元素对象代表dom元素，并提供了各种获取dom元素属性的方法。

使用您选择的任何编辑器在C:/> jsoup中创建以下Java程序。

JsoupTester.java

```
import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;
import org.jsoup.nodes.Element;

public class JsoupTester {
   public static void main(String[] args) {

      String html = "<html><head><title>Sample Title</title></head>"
         + "<body>"
         + "<p>Sample Content</p>"
         + "<div id='sampleDiv'><a href='www.google.com'>Google</a>"
         + "<h3><a>Sample</a><h3>"
         +"</div>"
         +"</body></html>";
      Document document = Jsoup.parse(html);

      //a with href
      Element link = document.select("a").first();

      System.out.println("Href: " + link.attr("href"));
   }
}
```

使用 javac 编译器编译类，如下所示:

```
C:\jsoup>javac JsoupTester.java
```

现在运行JsoupTester以查看输出。

```
C:\jsoup>java JsoupTester
```

查看输出。

```
Href: www.google.com
```

上一篇:jsoup - 使用选择器语法

下一篇:jsoup - 提取文本

jsoup - 提取文本介绍

下面的示例将HTML解析为Document对象后使用方法获取文本元素信息。

```
Document document=Jsoup.parse(html);
Element link=document.select("a").first();
System.out.println("Text: " + link.text());
```

元素对象代表dom元素，并提供各种方法来获取dom元素的文本。

Element.text 示例

使用您选择的任何编辑器在C:/> jsoup中创建以下Java程序。

JsoupTester.java

```
import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;
import org.jsoup.nodes.Element;

public class JsoupTester {
   public static void main(String[] args) {

      String html = "<html><head><title>Sample Title</title></head>"
         + "<body>"
         + "<p>Sample Content</p>"
         + "<div id='sampleDiv'><a href='www.google.com'>Google</a>"
         + "<h3><a>Sample</a><h3>"
         +"</div>"
         +"</body></html>";
      Document document = Jsoup.parse(html);

      //a with href
      Element link = document.select("a").first();

      System.out.println("Text: " + link.text());
   }
}
```

使用 javac 编译器编译类，如下所示:

```
C:\jsoup>javac JsoupTester.java
```

现在运行JsoupTester以查看输出。

```
C:\jsoup>java JsoupTester
```

查看输出。

```
Text: Google
```

上一篇:jsoup - 提取属性

下一篇:jsoup - 提取HTML

jsoup - 提取HTML介绍

以下示例将HTML解析为Document对象之后，使用html()和outerHtml方法获取html元素内容。

```
Document document=Jsoup.parse(html);
Element link=document.select("a").first();

System.out.println("Outer HTML: " + link.outerHtml());
System.out.println("Inner HTML: " + link.html());
```

元素对象代表dom元素，并提供各种方法来获取dom元素的html。

Element.html示例

使用您选择的任何编辑器在C:/> jsoup中创建以下Java程序。

JsoupTester.java

```
import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;
import org.jsoup.nodes.Element;

public class JsoupTester {
   public static void main(String[] args) {

      String html = "<html><head><title>Sample Title</title></head>"
         + "<body>"
         + "<p>Sample Content</p>"
         + "<div id='sampleDiv'><a href='www.google.com'>Google</a>"
         + "<h3><a>Sample</a><h3>"
         +"</div>"
         +"</body></html>";
      Document document = Jsoup.parse(html);

      //a with href
      Element link = document.select("a").first();

      System.out.println("Outer HTML: " + link.outerHtml());
      System.out.println("Inner HTML: " + link.html());
   }
}
```

使用 javac 编译器编译类，如下所示:

```
C:\jsoup>javac JsoupTester.java
```

现在运行JsoupTester以查看输出。

```
C:\jsoup>java JsoupTester
```

查看输出。

```
Outer HTML: <a href="www.google.com">Google</a>
Inner HTML: Google
```

上一篇:jsoup - 提取文本

下一篇:jsoup - 使用URL

jsoup - 使用URL介绍

下面的示例获取html页面中a元素的相对和绝对URL的方法。

```
String url="http://www.learnfk.com/";
Document document=Jsoup.connect(url).get();
Element link=document.select("a").first();

System.out.println("Relative Link: " + link.attr("href"));
System.out.println("Absolute Link: " + link.attr("abs:href"));
System.out.println("Absolute Link: " + link.absUrl("href"));
```

元素对象代表dom元素，并提供获取html页面中相对和绝对URL的方法。

Element.absUrl示例

使用您选择的任何编辑器在C:/> jsoup中创建以下Java程序。

JsoupTester.java

```
import java.io.IOException;

import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;
import org.jsoup.nodes.Element;

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

      String url = "http://www.learnfk.com/";
      Document document = Jsoup.connect(url).get();

      Element link = document.select("a").first();
      System.out.println("Relative Link: " + link.attr("href"));
      System.out.println("Absolute Link: " + link.attr("abs:href"));
      System.out.println("Absolute Link: " + link.absUrl("href"));
   }
}
```

使用 javac 编译器编译类，如下所示:

```
C:\jsoup>javac JsoupTester.java
```

现在运行JsoupTester以查看输出。

```
C:\jsoup>java JsoupTester
```

查看输出。

```
Relative Link: index.htm
Absolute Link: https://www.learnfk.com/index.htm
Absolute Link: https://www.learnfk.com/index.htm
```

上一篇:jsoup - 提取HTML

jsoup - 设置属性介绍

下面的示例将HTML解析为Document对象后，使用addClass或removeClass方法来增加或删除class类方法。

```
Document document=Jsoup.parse(html);
Element link=document.select("a").first();
link.attr("href","www.yahoo.com");
link.addClass("header");
link.removeClass("header");
```

元素对象代表dom元素，并提供了各种获取dom元素属性的方法。

addClass/removeClass示例

使用您选择的任何编辑器在C:/> jsoup中创建以下Java程序。

JsoupTester.java

```
import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;
import org.jsoup.nodes.Element;
import org.jsoup.select.Elements;

public class JsoupTester {
   public static void main(String[] args) {

      String html = "<html><head><title>Sample Title</title></head>"
        + "<body>"
        + "<p>Sample Content</p>"
        + "<div id='sampleDiv'><a id='googleA' href='www.google.com'>Google</a></div>"
        + "<div class='comments'><a href='www.sample1.com'>Sample1</a>"
        + "<a href='www.sample2.com'>Sample2</a>"
        + "<a href='www.sample3.com'>Sample3</a><div>"
        +"</div>"
        + "<div id='imageDiv' class='header'><img name='google' src='google.png?
imageView2/0/q/75|watermark/2/text/bGVhcm5may5jb20=/font/Y29uc29sYXM=/fontsize/400/fill/I0YxMTQxNA==/dissolve/100/gravity/SouthEast/dx/
10/dy/10' />"
        + "<img name='yahoo' src='yahoo.jpg?
imageView2/0/q/75|watermark/2/text/bGVhcm5may5jb20=/font/Y29uc29sYXM=/fontsize/400/fill/I0YxMTQxNA==/dissolve/100/gravity/SouthEast/dx/
10/dy/10' />"
        +"</div>"
        +"</body></html>";
      Document document = Jsoup.parse(html);

      //Example: set attribute
      Element link = document.getElementById("googleA");
      System.out.println("Outer HTML Before Modification :"  + link.outerHtml());
      link.attr("href","www.yahoo.com");
      System.out.println("Outer HTML After Modification :"  + link.outerHtml());
      System.out.println("---");

      //Example: add class
      Element div = document.getElementById("sampleDiv");
      System.out.println("Outer HTML Before Modification :"  + div.outerHtml());
      link.addClass("header");
      System.out.println("Outer HTML After Modification :"  + div.outerHtml());
      System.out.println("---");

      //Example: remove class
      Element div1 = document.getElementById("imageDiv");
      System.out.println("Outer HTML Before Modification :"  + div1.outerHtml());
      div1.removeClass("header");
      System.out.println("Outer HTML After Modification :"  + div1.outerHtml());
      System.out.println("---");

      //Example: bulk update
      Elements links = document.select("div.comments a");
      System.out.println("Outer HTML Before Modification :"  + links.outerHtml());
```

```
    links.attr("rel", "nofollow");
    System.out.println("Outer HTML Before Modification :"  + links.outerHtml());
  }
}
```

使用 javac 编译器编译类，如下所示:

```
C:\jsoup>javac JsoupTester.java
```

现在运行JsoupTester以查看输出。

```
C:\jsoup>java JsoupTester
```

查看输出。

```
Outer HTML Before Modification :<a id="googleA" href="www.google.com">Google</a>
Outer HTML After Modification :<a id="googleA" href="www.yahoo.com">Google</a>
---
Outer HTML Before Modification :<div id="sampleDiv">
 <a id="googleA" href="www.yahoo.com">Google</a>
</div>
Outer HTML After Modification :<div id="sampleDiv">
 <a id="googleA" href="www.yahoo.com" class="header">Google</a>
</div>
---
Outer HTML Before Modification :<div id="imageDiv" class="header">
 <img name="google" src="google.png?
imageView2/0/q/75|watermark/2/text/bGVhcm5may5jb20=/font/Y29uc29sYXM=/fontsize/400/fill/I0YxMTQxNA==/dissolve/100/gravity/SouthEast/dx/
10/dy/10">
 <img name="yahoo" src="yahoo.jpg?
imageView2/0/q/75|watermark/2/text/bGVhcm5may5jb20=/font/Y29uc29sYXM=/fontsize/400/fill/I0YxMTQxNA==/dissolve/100/gravity/SouthEast/dx/
10/dy/10">
</div>
Outer HTML After Modification :<div id="imageDiv" class="">
 <img name="google" src="google.png?
imageView2/0/q/75|watermark/2/text/bGVhcm5may5jb20=/font/Y29uc29sYXM=/fontsize/400/fill/I0YxMTQxNA==/dissolve/100/gravity/SouthEast/dx/
10/dy/10">
 <img name="yahoo" src="yahoo.jpg?
imageView2/0/q/75|watermark/2/text/bGVhcm5may5jb20=/font/Y29uc29sYXM=/fontsize/400/fill/I0YxMTQxNA==/dissolve/100/gravity/SouthEast/dx/
10/dy/10">
</div>
---
Outer HTML Before Modification :<a href="www.sample1.com">Sample1</a>
<a href="www.sample2.com">Sample2</a>
<a href="www.sample3.com">Sample3</a>
Outer HTML Before Modification :<a href="www.sample1.com" rel="nofollow">Sample1</a>
<a href="www.sample2.com" rel="nofollow">Sample2</a>
<a href="www.sample3.com" rel="nofollow">Sample3</a>
```

下一篇:jsoup - 设置HTML

jsoup - 设置HTML介绍

以下示例将HTML解析为Document对象之后，使用html，append，prepend()方法将值写入指定位置。

```
Document document=Jsoup.parse(html);
Element div=document.getElementById("sampleDiv");
div.html("<p>This is a sample content.</p>");
div.prepend("<p>Initial Text</p>");
div.append("<p>End Text</p>");
```

元素对象代表dom元素，并提供各种方法来将html设置，添加或添加到dom元素。

append/prepend/html示例

使用您选择的任何编辑器在C:/> jsoup中创建以下Java程序。

JsoupTester.java

```
import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;
import org.jsoup.nodes.Element;

public class JsoupTester {
   public static void main(String[] args) {

      String html = "<html><head><title>Sample Title</title></head>"
         + "<body>"
         + "<div id='sampleDiv'><a id='googleA' href='www.google.com'>Google</a></div>"
         +"</body></html>";
      Document document = Jsoup.parse(html);

      Element div = document.getElementById("sampleDiv");
      System.out.println("Outer HTML Before Modification :\n"  + div.outerHtml());
      div.html("<p>This is a sample content.</p>");
      System.out.println("Outer HTML After Modification :\n"  + div.outerHtml());
      div.prepend("<p>Initial Text</p>");
      System.out.println("After Prepend :\n"  + div.outerHtml());
      div.append("<p>End Text</p>");
      System.out.println("After Append :\n"  + div.outerHtml());
   }
}
```

使用 javac 编译器编译类，如下所示:

```
C:\jsoup>javac JsoupTester.java
```

现在运行JsoupTester以查看输出。

```
C:\jsoup>java JsoupTester
```

查看输出。

```
Outer HTML Before Modification :
<div id="sampleDiv">
 <a id="googleA" href="www.google.com">Google</a>
</div>
Outer HTML After Modification :
<div id="sampleDiv">
 <p>This is a sample content.</p>
</div>
After Prepend :
<div id="sampleDiv">
 <p>Initial Text</p>
 <p>This is a sample content.</p>
</div>
After Append :
<div id="sampleDiv">
 <p>Initial Text</p>
```

```
 <p>This is a sample content.</p>
 <p>End Text</p>
</div>
Outer HTML Before Modification :
<span>Sample Content</span>
Outer HTML After Modification :
<span>Sample Content</span>
```

上一篇:jsoup - 设置属性 下一篇:jsoup - 设置文本内容

jsoup - 设置文本内容介绍

下面的示例将HTML解析为Document对象之后，使用text，prepend，append方法将内容写入到指定元素内。

```
Document document=Jsoup.parse(html);
Element div=document.getElementById("sampleDiv");
div.text("This is a sample content.");
div.prepend("Initial Text.");
div.append("End Text.");
```

元素对象代表dom元素，并提供各种方法来将html设置，添加或添加到dom元素。

使用您选择的任何编辑器在C:/> jsoup中创建以下Java程序。

JsoupTester.java

```
import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;
import org.jsoup.nodes.Element;

public class JsoupTester {
   public static void main(String[] args) {

      String html = "<html><head><title>Sample Title</title></head>"
         + "<body>"
         + "<div id='sampleDiv'><a id='googleA' href='www.google.com'>Google</a></div>"
         +"</body></html>";
      Document document = Jsoup.parse(html);

      Element div = document.getElementById("sampleDiv");
      System.out.println("Outer HTML Before Modification :\n"  + div.outerHtml());
      div.text(This is a sample content.");
      System.out.println("Outer HTML After Modification :\n"  + div.outerHtml());
      div.prepend("Initial Text.");
      System.out.println("After Prepend :\n"  + div.outerHtml());
      div.append("End Text.");
      System.out.println("After Append :\n"  + div.outerHtml());
   }
}
```

使用 javac 编译器编译类，如下所示:

```
C:\jsoup>javac JsoupTester.java
```

现在运行JsoupTester以查看输出。

```
C:\jsoup>java JsoupTester
```

查看输出。

```
Outer HTML Before Modification :
<div id="sampleDiv">
 <a id="googleA" href="www.google.com">Google</a>
</div>
Outer HTML After Modification :
<div id="sampleDiv">
 This is a sample content.
</div>
After Prepend :
<div id="sampleDiv">
 Initial Text.This is a sample content.
</div>
After Append :
<div id="sampleDiv">
 Initial Text.This is a sample content.End Text.
</div>
```

上一篇:jsoup - 设置HTML

jsoup - 清理HTML介绍

以下示例将展示防止XSS攻击或跨站点脚本攻击的方法。

```
String safeHtml= Jsoup.clean(html, Whitelist.basic());
```

Jsoup对象使用白名单配置来清理html内容。

Jsoup.clean示例

使用您选择的任何编辑器在C:/> jsoup中创建以下Java程序。

JsoupTester.java

```
import org.jsoup.Jsoup;
import org.jsoup.safety.Whitelist;

public class JsoupTester {
   public static void main(String[] args) {

      String html = "<p><a href='http://example.com/'"
         +" onclick='checkData()'>Link</a></p>";

      System.out.println("Initial HTML: " + html);
      String safeHtml =  Jsoup.clean(html, Whitelist.basic());

      System.out.println("Cleaned HTML: " +safeHtml);
   }
}
```

使用 javac 编译器编译类，如下所示:

```
C:\jsoup>javac JsoupTester.java
```

现在运行JsoupTester以查看输出。

```
C:\jsoup>java JsoupTester
```

查看输出。

```
Initial HTML: <p><a href='http://example.com/' onclick='checkData()'>Link</a></p>
Cleaned HTML: <p><a href="http://example.com/" rel="nofollow">Link</a></p>
```