**코드 1.1 Jsoup 사용 준비**

|  |
| --- |
| <dependency>  <groupId>org.jsoup</groupId>  <artifactId>jsoup</artifactId>  <version>1.10.3</version>  </dependency> |

**코드 1.2 Jsoup로 만든 간단한 크롤러**

|  |
| --- |
| package kr.co.rint.crawler;  import org.jsoup.Jsoup;  import org.jsoup.nodes.Document;  import org.jsoup.nodes.Element;  import org.jsoup.select.Elements;  import java.nio.file.Files;  import java.nio.file.Paths;  public class SimpleCrawlerSample {    public static void main(String[] args) throws Exception {  // 위키북스 최상위 페이지  String url = "http://wikibook.co.kr/";  // GET 요청을 보내고 Document 객체를 변수 doc에 저장하기  Document doc = Jsoup.connect(url).get();  // CSS 선택자를 사용해 링크 추출하기  Elements elements = doc.select("li.book-in-front a");  // 반복문 적용하기  for(Element element: elements){  // 링크 내부의 글자 추출하기  String title = element.text().trim();  // 링크의 URL 추출하기  String nextUrl = element.attr("href");  // 링크 대상 페이지에 접근하기  Document nextDoc = Jsoup.connect(nextUrl).get();  // 책 소개 추출하기  String content = nextDoc.select("div.tab-content").html();  // "<책 제목>.html"이라는 이름으로 저장하기  Files.write(Paths.get(title + ".html"), content.getBytes("UTF-8"));  }  }  } |

**코드 1.3 crawler4j 사용 준비**

|  |
| --- |
| <dependency>  <groupId>edu.uci.ics</groupId>  <artifactId>crawler4j</artifactId>  <version>4.3</version>  </dependency> |

**코드 1.4 crawler4j로 만든 간단한 크롤러**

|  |
| --- |
| package kr.co.rint.crawler;  import edu.uci.ics.crawler4j.crawler.Page;  import edu.uci.ics.crawler4j.crawler.WebCrawler;  import edu.uci.ics.crawler4j.parser.HtmlParseData;  import edu.uci.ics.crawler4j.url.WebURL;  import org.jsoup.Jsoup;  import org.jsoup.nodes.Document;  public class MyCrawler extends WebCrawler {  @Override  public boolean shouldVisit(Page referringPage, WebURL url) {  // 크롤링해야 하는 페이지라면 True 리턴하기  // 위키북스의 모든 페이지 크롤링하기  String href = url.getURL();  return href.startsWith("http://wikibook.co.kr/");  }  @Override  public void visit(Page page) {  // 위키북스 사이트는 책을 따로 모아놓는 경로가 없으므로  // 예외 처리를 사용해 예외가 발생하지 않는 경우만 출력하기  // - 도서 페이지만 예외가 발생하지 않음  String url = page.getWebURL().getURL();  try {  HtmlParseData data = (HtmlParseData) page.getParseData();  // 페이지의 HTML을 파싱하기  Document doc = Jsoup.parse(data.getHtml());  // 타이틀 추출하기  String title = doc.select("#main-title").text();  // 책 내용 추출하기  String content = doc.select("div.tab-content").html();  // 타이틀, URL, 책 내용 출력하기  System.out.println(title + " - " + url);  System.out.println(content);  } catch(Exception e) {  }  }  } |

**코드 1.5 crawler4j로 만든 크롤러를 실행하는 클래스**

|  |
| --- |
| package kr.co.rint.crawler;  import edu.uci.ics.crawler4j.crawler.CrawlConfig;  import edu.uci.ics.crawler4j.crawler.CrawlController;  import edu.uci.ics.crawler4j.fetcher.PageFetcher;  import edu.uci.ics.crawler4j.robotstxt.RobotstxtConfig;  import edu.uci.ics.crawler4j.robotstxt.RobotstxtServer;  public class Crawler4jSample {  public static void main(String[] args) throws Exception {  // 크롤러 동시 실행 수 지정하기  int numberOfCrawlers = 1;    CrawlConfig config = new CrawlConfig();  // 시작 URL에서 몇 단계까지 들어갈지 설정하기  config.setMaxDepthOfCrawling(1);  // 크롤러의 데이터 저장 디렉터리 지정하기  config.setCrawlStorageFolder("./data/crawl/root");  // CrawlController 준비하기  PageFetcher pageFetcher = new PageFetcher(config);  RobotstxtConfig robotstxtConfig = new RobotstxtConfig();  RobotstxtServer robotstxtServer = new RobotstxtServer(robotstxtConfig, pageFetcher);  CrawlController controller = new CrawlController(config, pageFetcher, robotstxtServer);  // 크롤링 시작 URL 지정하기  controller.addSeed("http://takezoe.hatenablog.com/");  // 크롤링 시작하기  controller.start(MyCrawler.class, numberOfCrawlers);  }  } |

**코드 2.A 자바의 문자열 줄 바꿈**

|  |
| --- |
| System.out.print("LF 줄 바꿈\n");  System.out.print("CR 줄 바꿈\r");  System.out.print("CRLF 줄 바꿈\r\n"); |

**코드 2.1 요청과 응답 확인하기**

|  |
| --- |
| $ curl --verbose https://www.google.com  \* Rebuilt URL to: https://www.google.com/  \* Hostname was NOT found in DNS cache  \* Trying 172.217.26.100...  \* Connected to www.google.com (172.217.26.100) port 443 (#0)  \* TLS 1.2 connection using TLS\_ECDHE\_RSA\_WITH\_AES\_128\_CBC\_SHA  \* Server certificate: www.google.com  \* Server certificate: Google Internet Authority G2  \* Server certificate: GeoTrust Global CA  > GET / HTTP/1.1  > User-Agent: curl/7.37.1  > Host: www.google.com  > Accept: \*/\*  >  < HTTP/1.1 302 Foundf  < Cache-Control: private  < Content-Type: text/html; charset=UTF-8  < Location: https://www.google.co.kr/?gfe\_rd=cr&ei=AEvPWM61Ds2Q8Qet\_Ze4CA  < Content-Length: 262  < Date: Mon, 20 Mar 2017 03:22:40 GMT  < Alt-Svc: quic=":443"; ma=2592000; v="37,36,35"  <  <HTML><HEAD><meta http-equiv="content-type" content="text/html;➡  charset=utf-8">  <TITLE>302 Moved</TITLE></HEAD><BODY>  <H1>302 Moved</H1>  The document has moved  <A HREF="https://www.google.co.kr/?gfe\_rd=cr&amp;➡  ei=AEvPWM61Ds2Q8Qet\_Ze4CA">here</A>.  </BODY></HTML>  \* Connection #0 to host www.google.com left intact |

|  |
| --- |
| GET / HTTP/1.1  User-Agent: curl/7.37.1  Host: www.google.com  Accept: \*/\* |

**코드 2.2 PUT 메서드로는 요청을 전송할 수 없음**

|  |
| --- |
| <form method="PUT" action="/articles">  ...  </form> |

|  |
| --- |
| <form method="POST" action="/articles">  ...  <input type="hidden" name="\_method" value="PUT"/>  </form> |

|  |
| --- |
| POST /articles HTTP/1.1  Content-Type: application/json  X-HTTP-Method-Override: PUT  ... |

**코드 2.3 POST 메서드를 사용한 화면 이동**

|  |
| --- |
| <form action="/list" method="POST">  <input type="hidden" name="area" value="seoul"/>  <input type="hidden" name="page" value="1"/>  <input type="submit" name="previous" value="이전 페이지"/>  <input type="submit" name="next" value="다음 페이지"/>  </form> |

**코드 2.4 POST 요청 보내기**

|  |
| --- |
| Document doc = Jsoup.connect("http://www.example.com/list")  .data("area", "seoul")  .data("page", "1")  .data("next", "다음 페이지")  .post(); |

**코드 2.5 GET 메서드를 사용한 데이터 변경 처리**

|  |
| --- |
| <a href="/delete/item/123">제거</a> |

**코드 2.6 nofollow 속성으로 링크 대상을 순회하지 않게 만들기**

|  |
| --- |
| <a href="/delete/item/123" rel="nofollow">제거</a> |

|  |
| --- |
| Exception in thread "main" org.jsoup.HttpStatusException: HTTP error fetching URL. Status=404, URL=https://www.google.co.kr/123  at org.jsoup.helper.HttpConnection$Response.execute(HttpConnection.java:679)  at org.jsoup.helper.HttpConnection$Response.execute(HttpConnection.java:628)  at org.jsoup.helper.HttpConnection.execute(HttpConnection.java:260)  ... |

**코드 2.7 예외를 catch해서 상태 코드 확인하기**

|  |
| --- |
| String url = "https://www.google.co.kr/123";  try {  Response res = Jsoup.connect(url).execute();  ...  } catch (HttpStatusException ex){  int statusCode = ex.getStatusCode();  if(statusCode == 404){  System.out.println(url + "가 존재하지 않습니다.");  }  } |

**코드 2.8 예외를 throw하지 않게 만든 경우의 상태 코드 확인**

|  |
| --- |
| Response res = Jsoup.connect(url).ignoreHttpErrors(true).execute();  int statusCode = res.statusCode();  if(statusCode == 404){  System.out.println(url + "가 존재하지 않습니다.");  } |

|  |
| --- |
| Exception in thread "main" java.net.ConnectException: Connection refused  at java.net.PlainSocketImpl.socketConnect(Native Method)  at java.net.AbstractPlainSocketImpl.doConnect(AbstractPlainSocketImpl.java:345)  at java.net.AbstractPlainSocketImpl.connectToAddress(AbstractPlainSocketImpl.java:206)  at java.net.AbstractPlainSocketImpl.connect(AbstractPlainSocketImpl.java:188)  at java.net.SocksSocketImpl.connect(SocksSocketImpl.java:392)  at java.net.Socket.connect(Socket.java:589)  ... |

**코드 2.9 디폴트 리다이렉트**

|  |
| --- |
| Response res = Jsoup.connect("http://www.google.com/").execute();  System.out.println(res.url()); |

**코드 2.10 리다이렉트를 하고 싶지 않은 경우**

|  |
| --- |
| Response res = Jsoup.connect("http://www.google.com/").followRedirects(false).execute();  // 상태 코드 출력하기  int statusCode = res.statusCode();  System.out.println("Status: " + statusCode);  // Location 헤더 출력하기  String location = res.header("Location");  System.out.println("Location: " + location); |

**코드 2.11 HTML 내부의 meta 태그로 리다이렉트하기**

|  |
| --- |
| <!-- 5초 후에 http://www.example.com로 리다이렉트하기 -->  <meta http-equiv="refresh" content="5;URL=http://www.example.com"> |

**코드 2.12 meta 태그를 기반으로 리다이렉트 대상 URL 추출하기**

|  |
| --- |
| Document doc = Jsoup.connect("http://www.example.com/").get();  // meta 태그 추출하기  Elements elements = doc.select("meta[http-equiv=refresh]");  // content 속성의 값 추출하기  String value = elements.attr("content");  // content 속성의 값을 추출할 수 있다면 URL 추출하기  if(value.length() > 0){  // ";"로 분할하고, 이어서 "="로 분할해서 URL 부분 추출하기  String url = value.split(";")[1].split("=")[1].trim();  ...  } |

**코드 2.14 요청 헤더 설정과 응답 헤더 내용 확인**

|  |
| --- |
| Map<String, String> reqHeaders = new HashMap<>();  reqHeaders.put("User-Agent", "SampleCrawler");  Response res = Jsoup.connect("http://www.google.com/")  // 특정 요청 헤더 설정하기  .header("User-Agent", "SampleCrawler")  // Map<String, String>으로 한꺼번에 헤더 설정하기  .headers(reqHeaders)  // 일부 헤더는 추출 설정 메서드가 있음  .userAgent("SampleCrawler")  .execute();  // 특정 응답 헤더 추출하기  String value = res.header("Content-Type");  // 여러 헤더를 Map<String, String>으로 한꺼번에 추출하기  String contentType = res.contentType();  // 일부 헤더는 추출 전용 메서드가 있음  Map<String, String> resHeaders = res.headers(); |

**코드 2.15 독자적인 사용자 에이전트 설정하기(Jsoup의 경우)**

|  |
| --- |
| // 사용자 에이전트 설정하기(Jsoup의 경우)  Connection conn = Jsoup.connect(url);  conn.userAgent("MyCrawler"); |

**crawler4j의 사용자 에이전트**

|  |
| --- |
| crawler4j (https://github.com/yasserg/crawler4j/) |

독자적인 사용자 에이전트를 설정할 때는 코드 2.16처럼 합니다.

**코드 2.16 독자적인 사용자 에이전트 설정하기(crawler4j의 경우)**

|  |
| --- |
| // 사용자 에이전트 설정하기(crawler4j의 경우)  CrawlConfig config = new CrawlConfig();  config.setUserAgentString("MyCrawler"); |

**코드 2.17 Woothee를 사용해 사용자 에이전트가 크롤러의 것인지 판정하기**

|  |
| --- |
| import is.tagomor.woothee.Classifier;  String userAgent = ...  boolean isCrawler = Classifier.isCrawler(userAgent); |

**코드 2.18 Jsoup에서 이전 요청에서 응답받은 쿠키를 다음 요청에 사용하기**

|  |
| --- |
| // 처음 요청 전송하기  Response res = Jsoup.connect(url1).execute();  // 응답에서 쿠키 추출하기  String sessionId = res.cookie("JSESSIONID")  // 응답에서 추출한 쿠키를 넣어 요청 보내기  Document doc = Jsoup.connect(url2).cookie("JSESSIONID", sessionId).get(); |

|  |
| --- |
| Accept-Language: ko,en-US;q=0.8,en;q=0.6,ja;q=0.4 |

**코드 2.19 proxy() 메서드로 사용할 프록시 서버 지정하기**

|  |
| --- |
| Response res = Jsoup.connect("http://example.com/")  .proxy("127.0.0.1", 8080)  .method(Method.GET)  .execute(); |

**코드 2.20 시스템 속성으로 프록시 설정하기**

|  |
| --- |
| // HTTP의 경우  System.setProperty("http.proxyHost", "127.0.0.1");  System.setProperty("http.proxyPort", "8080");  System.setProperty("http.proxyUser", "username");  System.setProperty("http.proxyPassword", "password");  // HTTPS의 경우  System.setProperty("https.proxyHost", "127.0.0.1");  System.setProperty("https.proxyPort", "8080");  System.setProperty("https.proxyUser", "username");  System.setProperty("https.proxyPassword", "password"); |

**코드 2.21 자바 VM 실행 시 시스템 속성 지정하기**

|  |
| --- |
| $ java -Dhttp.proxyHost=127.0.0.1 -Dhttp.proxyPort=8080 kr.co.rint.crawler.SampleCrawler |

|  |
| --- |
| $ curl -x 127.0.0.1:8080 -XGET http://www.example.com/ |

**코드 2.22 키 저장소에 인증서 추가하기**

|  |
| --- |
| keytool -importcert -v -trustcacerts -file /path/to/cert.crt -keystore $JAVA\_HOME/jre/lib/security/cacerts |

**코드 2.23 SSL 인증서 검증 비활성화하기**

|  |
| --- |
| Response res = Jsoup.connect("https://example.com/")  .validateTLSCertificates(false) // SSL 인증서 검증 비활성화하기  .execute(); |

**코드 3.1 HTTP 응답의 Content-Type**

|  |
| --- |
| HTTP/1.1 200 OK  Date: Sun, 23 Oct 2016 09:00:00 GMT  Server: Apache  Accept-Ranges: bytes  Vary: Accept-Encoding,User-Agent  Content-Encoding: gzip  Content-Length: 4051  Connection: close  Content-Type: text/html; charset=euc\_kr |

**코드 3.2 meta 태그에 문자 코드 정보가 지정된 경우**

|  |
| --- |
| <head>  ...  <meta http-equiv="Content-Type" content="text/html;charset=shift\_jis">  <meta charset="shift\_jis">  ...  </head> |

**코드 3.3 텍스트 파일을 읽고 쓸 때 문자 깨짐이 일어나지 않게 하기**

|  |
| --- |
| int ch;  File file = new File("./hello.txt");  // 문자 깨짐을 일으킬 가능성이 있는 처리 예  // FileReader는 파일이 디폴트 문자 코드로 복호화됐다는 전제하에 파일을 읽어 들입니다.  // 따라서 파일의 문자 코드와 실제 환경의 디폴트 문자 코드가 다르면  // 문자 깨짐이 일어납니다.  FileReader fr = new FileReader(file);  while ((ch = fr.read()) != -1) {  System.out.println((char)ch);  }  fr.close();  // InputStreamReader에 문자 코드를 명시적으로 지정해  // 디폴트 문자 코드에 의존하지 않게 만듭니다.  InputStream is = new FileInputStream(file);  InputStreamReader isr = new InputStreamReader(is, "EUC-KR");  while ((ch = isr.read()) != -1) {  System.out.println((char)ch);  }  isr.close(); |

**코드 3.4 Jsoup가 판정한 문자 코드를 추출하는 예**

|  |
| --- |
| String url = "http://www.example.com";  Response response = Jsoup.connect(url).execute();  System.out.println("HTTP 응답을 받았습니다.");  // HTTP 응답 헤더에 문자 코드가 포함된 경우  // Connection.execute() 메서드 실행 후에 문자 코드 이름을 추출할 수 있습니다.  System.out.println("문자 코드는 " + response.charset() + "입니다.");  // HTTP 응답 헤더에 문자 코드가 포함돼 있지 않고,  // HTML 내부의 meta 태그에 문자 코드 정보가 포함된 경우  // Response.parse() 메서드 실행 후에 문자 코드 이름을 추출할 수 있습니다.  response.parse();  System.out.println("응답을 파싱했습니다.");  System.out.println("문자 코드는 " + response.charset() + "입니다."); |

**코드 3.5 HTTP 통신 이후 정규 표현식으로 문자 코드 정보 추출하기**

|  |
| --- |
| // Content-Type 또는 meta 태그에서 문자 코드 이름을 추출하는 정규 표현식  Pattern charsetPattern = Pattern.compile("(?i) bcharset= s\*(?: "|')?➡ ([^ s,; "']\*)");  URL url = new URL("http://example.com");  HttpURLConnection connection = (HttpURLConnection) url.openConnection();  connection.setRequestMethod("GET");  connection.connect();  // Content-Type 헤더에서 문자 코드 추출하기  // "text/html; charset=euc\_kr" 등의 문자열 리턴  String contentType = connection.getContentType();  Matcher contentTypeMatcher = charsetPattern.matcher(contentType);  if (contentTypeMatcher.find()) {  String charsetName = contentTypeMatcher.group(1).trim();  // Content-Type에 있는 문자 코드 출력하기  System.out.println("Charset in Content-Type: " + charsetName);  }  // meta 태그에서 문자 코드 정보 추출하기  // W3C 문서에 따르면 문자 코드 정보는  // 파일 앞부터 1024바이트까지 내부에 포함돼 있어야 하므로  byte[] first1k = new byte[1024];  BufferedInputStream in = new BufferedInputStream(connection.getInputStream());  in.read(first1k);  in.close();  Matcher metaMatcher = charsetPattern.matcher(new String(first1k, "UTF-8"));  if(metaMatcher.find()) {  String charsetName = metaMatcher.group(1).trim();  // meta charset에 있는 문자 코드 출력하기  System.out.println("Charset in HTML: " + charsetName);  }  connection.disconnect(); |

**코드 3.6 일반 문자열을 결합 문자열로 변환하고 다시 결합 문자열을 일반 문자열로 변환하기**

|  |
| --- |
| // 원본 문자열  String combined = "무제 폴더";  // NFD 형식으로 변환합니다.  String nfdNormalized = java.text.Normalizer.normalize(combined, java.text.Normalizer.Form.NFD);  System.out.println(nfdNormalized);  // NFC 형식으로 변환합니다.  String nfcNormalized = java.text.Normalizer.normalize(nfdNormalized,  java.text.Normalizer.Form.NFC);  System.out.println(nfcNormalized); |

**코드 3.7 새니타이즈**

|  |
| --- |
| String unsafeHtml =  "<a href='javascript:alert( "Oops! ")'>Hello!</a>" +  "<script>alert( "Ouch! ");</script>";  String sanitizedHtml = Jsoup.clean(unsafeHtml, Whitelist.basic()); ➡  // 화이트 리스트 방식으로 새니타이즈  System.out.println(sanitizedHtml); // <a rel="nofollow">Hello!</a> |

**코드 3.8 새니타이즈 후에 정규화하면 위험!**

|  |
| --- |
| String unsafeHtml = "＜script＞alert( "hello ");＜/script＞";  // "＜"와 "＞"라서 HTML 태그로 인식하지 않음  String sanitizedHtml = Jsoup.clean(unsafeHtml, Whitelist.basic());  // 정규화 하면 "＜"와 "＞"가 "<"와 ">"로 변함  String normalizedHtml = Normalizer.normalize(sanitizedHtml, Form.NFKD);  // 결과적으로 "<script>alert("hello")</script>"라고 변경돼 위험한 문자열이 됨  System.out.println(normalizedHtml); |

**코드 3.9 문자 코드에서 Charset 객체를 추출할 때 실패하는 경우**

|  |
| --- |
| String charsetName = ...  Charset charset = Charset.forName(charsetName); // 문자 코드 이름으로  // 객체 생성하기  byte[] responseBody = ... // 응답 바디의 바이트  String decodedBody = new String(responseBody, charset); // 디코드된 문자열 |

.

**코드 3.10 문자 코드 이름을 기반으로 Charset 객체 생성하기**

|  |
| --- |
| // 문자 코드 이름으로 유효한 문자열을 나타내는 정규 표현식  static Pattern charsetNamePattern = Pattern.compile("^[a-zA-Z0-9[^ -+:\_.]][a-zA-Z0-9 -+:\_.]+$");  public Charset getCharset(String charsetName) {  Charset charset;  if (charsetName != null) {  // 전달된 문자열에서 문자 코드 이름이 유효한지,  // 지원하는 문자 코드인지 확인하기  Matcher matcher = charsetNamePattern.matcher(charsetName);  if (matcher.matches() && Charset.isSupported(charsetName)) {  charset = Charset.forName(charsetName);  } else {  // 유효하지 않거나 지원하지 않는 경우는  // 디폴트 문자 코드로 UTF-8 사용하기  charset = StandardCharsets.UTF\_8;  }  } else {  // 문자 코드 이름이 null이라면 UTF-8 사용하기  charset = StandardCharsets.UTF\_8;  }  return charset;  }  ...  // HTTP 헤더의 Content-Type에서 추출한 문자 코드 이름  String charsetNameFromContentType = ...  // Charset 객체 생성하기  Charset charset = getCharset(charsetNameFromContentType); |

**코드 3.12 MySQL 5.6에서 텍스트를 삽입할 때 실패하는 예**

|  |
| --- |
| mysql> CREATE TABLE text\_with\_emoji (text varchar(255)) CHARSET utf8;  Query OK, 0 rows affected (0.03 sec)  mysql> INSERT INTO text\_with\_emoji VALUES ('오늘 저녁은 ###[이모티콘 추가(112쪽)]### 먹으러 왔다!');  Query OK, 1 row affected, 2 warnings (0.01 sec)  mysql> SELECT \* FROM text\_with\_emoji;  +-------------+  | text |  +-------------+  | 오늘 저녁은 |  +-------------+  1 row in set (0.00 sec) |

**코드 3.13 MySQL 5.6에서 문자 변환에 실패할 경우 오류로 처리하기(my.conf)**

|  |
| --- |
| [mysqld]  sql\_mode='NO\_ENGINE\_SUBSTITUTION,STRICT\_ALL\_TABLES' |

**코드 3.14 Charset이 utf8mb4로 설정된 테이블에 그림 문자를 넣은 경우의 문제**

|  |
| --- |
| mysql> SET NAMES utf8mb4;  Query OK, 0 rows affected (0.00 sec)  mysql> CREATE TABLE emoji\_list (emoji varchar(1), name varchar(5)) ➡  CHARSET utf8mb4;  Query OK, 0 rows affected (0.02 sec)  mysql> INSERT INTO emoji\_list VALUES ('###[이모티콘 추가(코드내 이모티콘 모두 원서 114쪽)]###', '초밥'), ('###[이모티콘 추가]###', '맥주');  Query OK, 2 rows affected (0.00 sec)  Records: 2 Duplicates: 0 Warnings: 0  mysql> select \* from emoji\_list where emoji = '###[이모티콘 추가]###';  +-------+---------+  | emoji | name |  +-------+---------+  |[추가] | 초밥 |  |[추가] | 맥주 |  +-------+---------+  2 rows in set (0.00 sec) |

**코드 3.15 특정 컬럼의 대조 확인 순서를 utf8mb4\_bin으로 지정하기**

|  |
| --- |
| mysql> CREATE TABLE emoji\_list\_bin (emoji varchar(1) COLLATE utf8mb4\_bin, name varchar(5)) CHARSET utf8mb4;  Query OK, 0 rows affected (0.02 sec)  mysql> INSERT INTO emoji\_list\_bin VALUES ('[이모티콘 추가]', '초밥'), ('[이모티콘 추가]', '맥주');  Query OK, 2 rows affected (0.00 sec)  Records: 2 Duplicates: 0 Warnings: 0  mysql> SELECT \* FROM emoji\_list\_bin WHERE emoji = '[이모티콘 추가]';  +-------+--------+  | emoji | name |  +-------+--------+  |[추가(115쪽)] | 초밥 |  +-------+--------+  1 row in set (0.00 sec) |

**코드 3.16 SELECT 시 COLLATE 구문으로 utf8mb4\_bin 지정하기**

|  |
| --- |
| mysql> SELECT \* FROM emoji\_list WHERE emoji = '[추가]';  +-------+---------+  | emoji | name |  +-------+---------+  |[추가(116쪽)] | 초밥 |  |[추가] | 맥주 |  +-------+---------+  2 rows in set (0.00 sec)  mysql> SELECT \* FROM emoji\_list WHERE emoji = '[추가]' COLLATE utf8mb4\_bin;  +-------+---------+  | emoji | name |  +-------+---------+  |[추가] | 초밥 |  +-------+---------+  1 row in set (0.00 sec) |

**코드 3.19 -C 옵션을 붙여 바이트열에 해당하는 ASCII 문자 출력하기**

|  |
| --- |
| $ echo 안녕하세요, hexdump[이모티콘 추가(123쪽)]! | hexdump -C  00000000 e3 81 93 e3 82 93 e3 81 ab e3 81 a1 e3 81 af e3 |................|  00000010 80 81 20 68 65 78 64 75 6d 70 f0 9f 98 80 21 0a |.. hexdump....!.|  00000020 |

**코드 3.20 문자 깨짐이 일어나는 웹 사이트 조사하기**

|  |
| --- |
| $ curl http://example.com | hexdump -C |

**코드 3.21 curl 명령어로 출력 결과를 파일로 저장하기**

|  |
| --- |
| $ curl http://example.com > index.html |

**코드 3.22 HTTP 응답**

|  |
| --- |
| // HTTP 헤더에 부적절한 문자 코드 이름이 지정돼 있는 예  // 샘플 코드를 붙여넣어 만들었는지 "문자 코드"라는 문자열이  // URL 인코드된 charset 매개 변수에 지정돼 있음  Content-Type: text/html;charset=%BB%38%C7%90%20%CF%54%B4%DC |

**코드 3.23 HTML 내부의 meta 태그**

|  |
| --- |
| <!--  HTML에 부적절한 문자 코드 이름이 지정된 예  프로그램 실수인지 None이라는 문자열이 들어 있음  -->  <meta charset="None"> |

**코드 3.24 juniversalchardet를 사용해 문자 코드 판별하기**

|  |
| --- |
| /\*\*  \* InputStream 바이트열을 기반으로 문자 코드를 추출하는 메서드  \* @param in 문자 코드 추정에 사용할 InputStream  \* @return 추정된 문자 코드 이름  \* @throws IOException  \*/  public static String detectCharsetName(InputStream in) throws IOException {  UniversalDetector detector = new UniversalDetector(null);  int mark;  byte[] buf = new byte[1024];  while ((mark = in.read(buf)) > 0 && !detector.isDone()) {  detector.handleData(buf, 0, mark);  }  detector.dataEnd();  return detector.getDetectedCharset();  } |

**코드 3.25 Jsoup와 조합해 사용하기**

|  |
| --- |
| static Pattern charsetAttrPattern = Pattern.compile("(?i)\\bcharset=\\s\*(?:\"|')?([^\\s,;\"']\*)");  static Pattern charsetNamePattern = Pattern.compile("^[a-zA-Z0-9[^\\-+:\_.]][a-zA-Z0-9\\-+:\_.]+$");  /\*\*  　\* 바이트열을 기반으로 문자 코드 이름 추정하기  　\*/  public static String detectCharsetName(byte[] bytes) throws IOException {  InputStream in = new ByteArrayInputStream(bytes);  UniversalDetector detector = new UniversalDetector(null);  int mark;  byte[] buf = new byte[1024];  while ((mark = in.read(buf)) > 0 && !detector.isDone()) {  detector.handleData(buf, 0, mark);  }  detector.dataEnd();  return detector.getDetectedCharset();  }  /\*\*  　\* 전달된 문자 코드 이름이 문자 코드 이름으로 valid한지 확인하기  　\*/  public static Charset validateCharset(String charsetName) {  Charset charset;  if (charsetName != null) {  // 전달된 문자열이 문자 코드 이름으로 유효한지,  // 지원하는 문자 코드 이름인지 판정하기  Matcher matcher = charsetNamePattern.matcher(charsetName);  if (matcher.matches() && Charset.isSupported(charsetName)) {  charset = Charset.forName(charsetName);  } else {  // 유효하지 않거나 지원하지 않는 경우는  // 디폴트 문자 코드로 UTF-8 사용하기  charset = StandardCharsets.UTF\_8;  }  } else {  // 문자 코드 이름이 null이라면 UTF-8 사용하기  charset = StandardCharsets.UTF\_8;  }  return charset;  }  public static void main(String args[]) throws IOException {  String url = "http://example.com";  Connection.Response response = Jsoup.connect(url).execute();    Document doc;  String foundCharsetName = response.charset();    if (foundCharsetName != null) {  doc = response.parse();  } else {  Document tmpDoc = response.parse();  Element meta = tmpDoc.select("meta[http-equiv=content-type], meta[charset]").first();  if (meta != null) { // 문자 코드 정보를 포함한 요소가 있는 경우  // 해당 요소를 기반으로 문자 코드 추출하기  if (meta.hasAttr("charset")) {  foundCharsetName = meta.attr("charset");  } else if (meta.hasAttr("http-equiv")) {  Matcher m = charsetAttrPattern.matcher(meta.attr("content"));  if (m.find()) {  foundCharsetName = m.group(1).trim().replace("charset=", "");  }  }  } else { // 문자 코드 정보를 포함한 요소가 없는 경우  // 바이트열을 기반으로 문자 코드 추정하기  foundCharsetName = detectCharsetName(response.bodyAsBytes());  }  Charset foundCharset = validateCharset(foundCharsetName);  response.charset(foundCharset.name());  doc = response.parse();  }  System.out.println(doc.body());  } |

**코드 3.26 ICU4J로 문자 코드를 판정하는 메서드**

|  |
| --- |
| /\*\*  \* 바이트열을 기반으로 문자 코드 추정합니다. 추정 정밀도 점수가 50점 이상인 경우  \* 추정한 문자 코드 결과를 리턴합니다.  \* 추정할 수 없는 경우, 또는 정밀도 점수 50점 이상이 없는 경우는  \* 디폴트 문자 코드로 UTF-8을 리턴합니다.  \* @param in 문자 코드 추정에 사용할 바이트열  \* @return 추정한 문자 코드에 해당하는 Charset 객체  \*/  public static Charset detectCharset(byte[] in) {  CharsetDetector detector = new CharsetDetector();  detector.setText(in);  CharsetMatch result = detector.detect();  if (result != null) {  int confidence = result.getConfidence(); // 추정 정밀도(0~100) 추출하기  String detectedCharsetName = result.getName();  return (confidence >= 50) ? Charset.forName(detectedCharsetName) : StandardCharsets.UTF\_8;  } else {  return StandardCharsets.UTF\_8;  }  } |

**코드 4.1 정규 표현식으로 HTML 내부의 모든 a 태그 추출하기**

|  |
| --- |
| String html = ...  // a 태그를 추출하기 위한 정규 표현  Pattern regex = Pattern.compile("<a.\*>.\*?</a>");  // Matcher 객체 생성하기  Matcher matcher = regex.matcher(html);  // 정규 표현식에 일치하는 부분은 콘솔에 출력하기  while(matcher.find()){  System.out.println(matcher.group());  } |

|  |
| --- |
| <?xml version="1.0"?>  <books>  <book>  <title>크롤링</title>  <publisher>위키북스</publisher>  </book>  ...  </books> |

|  |
| --- |
| /books/book[title/text()='크롤링']/publisher |

**코드 4.2 XPath로 스크레이핑하기**

|  |
| --- |
| // XML 파일을 읽어 들입니다.  DocumentBuilderFactory documentBuilderFactory = DocumentBuilderFactory.newInstance();  DocumentBuilder documentBuilder = documentBuilderFactory.newDocumentBuilder();  Document doc = documentBuilder.parse(new File("books.xml"));  // XPath 사용 준비를 합니다.  XPathFactory xpathFactory = XPathFactory.newInstance();  XPath xpath = xpathFactory.newXPath();  // publisher 요소를 추출하기 위한 XPath를 생성합니다.  XPathExpression expr = xpath.compile("/books/book[title/text()='크롤링']/publisher");  // publisher 요소를 추출하고 콘솔에 출력합니다.  Object result = expr.evaluate(doc, XPathConstants.NODE);  Element element = (Element) result;  System.out.println(element.getTextContent()); |

**코드 4.A pom.xml에 TagSoup 의존 관계 추가하기**

|  |
| --- |
| <dependency>  <groupId>org.ccil.cowan.tagsoup</groupId>  <artifactId>tagsoup</artifactId>  <version>1.2.1</version>  </dependency> |

**코드 4.B HTML을 XML로 변환하기**

|  |
| --- |
| StringWriter out = new StringWriter();  // HTML 파일을 읽어 들인 뒤 XML로 변환하기  Parser parser = new Parser();  parser.setContentHandler(new XMLWriter(out));  parser.parse(new InputSource(new FileInputStream("test.html")));  // 변환 후의 XML을 문자열로 추출하기  String xml = out.toString(); |

|  |
| --- |
| h1 {  font-size: 180%;  color: red;  }  div.content {  padding: 10px;  color: gray;  } |

**코드 4.3 pom.xml에 Apache Tika의 의존 관계 추가하기**

|  |
| --- |
| <dependency>  <groupId>org.apache.tika</groupId>  <artifactId>tika-parsers</artifactId>  <version>1.14</version>  </dependency> |

**코드 4.4 Apache Tika로 PDF 파일에서 텍스트 추출하기**

|  |
| --- |
| // Tika 사용 준비하기  Tika tika = new Tika();  // PDF 파일에서 텍스트를 추출해 콘솔에 출력하기  String result = tika.parseToString(new File("sample.pdf"));  System.out.println(result); |

|  |
| --- |
| <table>  <tbody>  <tr>  <th>이름</th>  <td>가방</td>  </tr>  <tr>  <th>가격</th>  <td>98000원</td>  </tr>  </tbody>  </table> |

|  |
| --- |
| tr:nth-child(2) th + td |

|  |
| --- |
| <table>  <tbody>  <tr>  <th>이름</th>  <td>플레어 스커트</td>  </tr>  <tr>  <th>색</th>  <td>흰색</td>  </tr>  <tr>  <th>가격</th>  <td>30000원</td>  </tr>  </tbody>  </table> |

|  |
| --- |
| <table>  <tbody>  <tr>  <th>이름</th>  <td>가방</td>  </tr>  <tr>  <th>가격</th>  <td>98000원</td>  </tr>  </tbody>  </table> |

|  |
| --- |
| th:contains(가격) + td |

|  |
| --- |
| <tr>  <th>가격</th>  <td>98000원</td>  </tr> |

|  |
| --- |
| <tr>  <th>판매가</th>  <td>98000원</td>  </tr> |

|  |
| --- |
| th:matches(가격|판매가) + td |

|  |
| --- |
| <tr>  <td>가격</td>  <td>98000원</td>  <td><a href="http://example.com">주문하기</a></td>  </tr> |

|  |
| --- |
| td:contains(가격) + td |

|  |
| --- |
| <tr>  <td>가격</td>  <td>88000원(가격 조정 가능)</td>  <td><a href="http://example.com">주문하기</a></td>  </tr> |

|  |
| --- |
| td:matches(^가격$) + td |

|  |
| --- |
| <tr>  <td>색</td>  <td>붉은색<span>※색 견본 있음</span></td>  <td>10000원</td>  </tr> |

|  |
| --- |
| td:containsOwn(색) + td |

|  |
| --- |
| <h1>한강 벚꽃 축제</h1>  <div>  <p>2017년 04월 20일</p>  <p style="font-size: 12px;">2017년 04월 20일부터 한강에서 벚꽃 축제가 열렸습니다....</p>  </div> |

|  |
| --- |
| p[style] |

**Microdata의 사용 예**

|  |
| --- |
| <div itemscope itemtype="http://schema.org/Article" itemprop="mainEntity">  <h1 itemprop="headline">한강 벚꽃 축제</h1>  <p itemprop="datePublished">2017/04/10</p>  <p itemprop="articleBody">2017년 04월 20일부터 한강에서 벚꽃 축제가 열렸습니다....</p>  </div> |

|  |
| --- |
| <h1 itemprop="headline">한강 벚꽃 축제</h1> |
| h1[itemprop="headline"] |

|  |
| --- |
| <h1 class="title" itemprop="headline">한강 벚꽃 축제</h1> |
| h1[class="title"][itemprop="headline"] |

|  |
| --- |
| <h1 class="title main" itemprop="headline">한강 벚꽃 축제</h1> |
| h1[class ~="main"] |

|  |
| --- |
| <div>  <img src="img/large-image-1111.png">  <img src="img/small-image-1111.png">  <img src="img/large-image-1112.png">  <img src="img/small-image-1112.png">  <img src="img/icon.png">  </div> |

|  |
| --- |
| img[src~=img/(large|small)-image-[0-9]+.png] |

|  |
| --- |
| <a href="/job-detail.html?id=111">채용 정보</a>  <a href="/company-detail.html?id=111">기업 정보</a> |
| a[href^=/job-detail] |

|  |
| --- |
| <img src="image.png">  <img src="image.jpeg"> |
| img[src$=.jpeg] |

|  |
| --- |
| <img src="large-image-A.png">  <img src="large-image-B.png">  <img src="small-image-A.png">  <img src="small-image-B.png"> |
| img[src\*=image-A] |

|  |
| --- |
| <link rel="alternate" href="http://example.com/english/index.html" hreflang="en" />  <link rel="alternate" href="http://example.com/english-us/index.html" hreflang="en-us" />  <link rel="alternate" href="http://example.com/english-gb/index.html" hreflang="en-gb" /> |
| link[hreflang|="en"] |

|  |
| --- |
| a:not([href\*="^"]) |

|  |
| --- |
| a[href~=^(?!.\* ^).+$] |

|  |
| --- |
| <img src="img/size-s.jpeg" class="icon-size" alt="S 사이즈">  <img src="img/size-m.jpeg" class="icon-size" alt="M 사이즈">  <img src="img/size-l.jpeg" class="icon-size" alt="L 사이즈"> |

**코드 4.5 Jsoup에서 alt 속성의 데이터를 추출한 뒤 구분하는 예**

|  |
| --- |
| // alt 속성을 가진 img 요소 추출하기  Elements elements = doc.select("img.size-icon[alt]");  for(Element e: elements){  String size = null;    // alt 속성의 값을 추출하고 적절한 구분 값으로 변환하기  String alt = e.attr("alt");  switch(alt){  case "S 사이즈": size = "S"; break;  case "M 사이즈": size = "M"; break;  case "L 사이즈": size = "L"; break;  default: size = "N";  }    // 변환 결과를 콘솔에 출력하기  System.out.println(size);  } |

|  |
| --- |
| https://maps.googleapis.com/maps/api/geocode/json?address=%EA%B2%BD%EA%B8%B0%EB%8F%84%20%ED%8C%8C%EC%A3%BC%EC%8B%9C%20%EB%AC%B8%EB%B0%9C%EB%A1%9C%20115%20%EC%84%B8%EC%A2%85%EC%B6%9C%ED%8C%90%EB%B2%A4%EC%B2%98%ED%83%80%EC%9A%B4%20311%ED%98%B8 |

**코드 4.8 meta 태그로 웹 페이지의 메타 데이터 정의하기**

|  |
| --- |
| <head>  <title>웹 크롤링과 스크레이핑</title>  ...  <meta name="description" content=웹 사이트를 크롤링하는 실전적인 방법을 다루는 도서입니다 ">  <meta name="keywords" content="크롤러, 크롤링, 스크레이핑">  ...  </head> |

**코드 4.9 HTML에 포함된 PageMap의 예**

|  |
| --- |
| <!--  <PageMap>  <DataObject type="action">  <Attribute name="label" value="Download"/>  <Attribute name="url" value="http://www.scribd.com/document\_downloads/20258723?extension=pdf"/>  <Attribute name="class" value="Download"/>  </DataObject>  <DataObject type="action">  <Attribute name="label" value="Fullscreen View"/>  <Attribute name="url" value="http://d1.scribdassets.com/➡  ScribdViewer.swf?document\_id=20258723&access\_key=key-27lwdyi9z21ithon73g3&version=1&viewMode=fullscreen"/>  <Attribute name="class" value="fullscreen"/>  </DataObject>  </PageMap>  --> |

**코드 4.10 PageMap을 추출하는 프로그램의 예**

|  |
| --- |
| public static void main(String[] args) throws Exception {  // HTML 파싱하기  Document doc = Jsoup.parse(...);  // HTML 내부의 모든 노드를 재귀적으로 처리하기  processNode(doc);  }  private static void processNode(Node node){  for(Node child: node.childNodes()){  if(child instanceof Comment){  // 주석일 경우  Comment comment = (Comment) child;  if(comment.getData().trim().startsWith("<PageMap>")){  // 주석의 내용이 PageMap일 경우  parsePageMap(comment.getData());  }  } else {  // 주석 이외의 노드라면 계속 재귀적으로 처리하기  processNode(child);  }  }  }  private static void parsePageMap(String comment){  // PageMap 파싱하기  Document pageMap = Jsoup.parse(comment);  // CSS 선택자로 메타 데이터 추출해서 출력하기  Elements elements = pageMap.select("DataObject[type=thumbnail]>Attribute[name=src]");  System.out.println(elements.attr("value"));  } |

**코드 4.11 OGP**

|  |
| --- |
| <head prefix="og:http://ogp.me/ns#">  <meta property="og:title" content="한강 벚꽃 축제" />  <meta property="og:description" content="2017년 04월 20일부터 한강에서 벚꽃 축제가 열렸습니다." />  <meta property="og:type" content="article" />  <meta property="og:url" content="http://example.com/news/001.html" />  <meta property="og:image" content="http://example.com/sample.jpg" />  </head> |

**코드 4.12 OGP를 사용해 같은 속성 여러 개 입력하기**

|  |
| --- |
| <meta property="og:image" content="http://example.com/sample1.jpg" />  <meta property="og:image" content="http://example.com/sample2.jpg" /> |

**코드 4.13 OGP를 사용해 각각의 이미지에 구조화 설정 지정하기**

|  |
| --- |
| <meta property="og:image" content="http://example.com/sample1.jpg" />  <meta property="og:image:width" content="300" />  <meta property="og:image:height" content="300" />  <meta property="og:image" content="http://example.com/sample2.jpg" />  <meta property="og:image:width" content="500" />  <meta property="og:image:height" content="500" />  <meta property="og:image" content="http://example.com/sample3.jpg" /> |

**코드 4.14 Twitter Card**

|  |
| --- |
| <meta name="twitter:card" content="summary\_large\_image" />  <meta name="twitter:image" content="http://example.com/sample.jpg" />  <meta name="twitter:title" content="한강 벚꽃 축제" />  <meta name="twitter:description" content="2017년 04월 20일부터 한강에서 벚꽃 축제가 열렸습니다." /> |

**코드 4.15 타이틀이 p 태그로 만들어진 웹 페이지**

|  |
| --- |
| <div>  <p style="color:#0004ff; font-size:150%;">한강 벚꽃 축제</p>  <p>2017/04/10</p>  <p>2017년 04월 20일부터 한강에서 벚꽃 축제가 열렸습니다....</p>  </div> |

**코드 4.16 Microformats**

|  |
| --- |
| <div class="hentry">  <p class="entry-title">한강 벚꽃 축제</p>  <p class="published">2017/04/10</p>  <p class="entry-content">2017년 04월 20일부터 한강에서 벚꽃 축제가 열렸습니다....</p>  </div> |

**코드 4.17 Microdata**

|  |
| --- |
| <div itemscope itemtype="http://schema.org/Article" itemprop="mainEntity">  <p itemprop="headline">한강 벚꽃 축제</p>  <p itemprop="datePublished">2017/04/10</p>  <p itemprop="articleBody">2017년 04월 20일부터 한강에서 벚꽃 축제가 열렸습니다....</p>  <!-- 일부 필수 요소 생략 -->  </div> |

**코드 4.18 Microdata의 itemscope 속성**

|  |
| --- |
| <div itemscope itemtype="http://schema.org/Article">  <!-- 생략 -->  </div> |

**코드 4.19 Microdata의 itemtype 속성**

|  |
| --- |
| <div itemscope itemtype="http://schema.org/Article">  <!-- 생략 -->  </div> |

**코드 4.20 Microdata의 itemprop 속성**

|  |
| --- |
| <div itemscope itemtype="http://schema.org/Article" itemprop="mainEntity">  <h1 itemprop="headline">한강 벚꽃 축제</h1>  <p itemprop="datePublished">2017/04/10</p>  <p itemprop="articleBody">2017년 04월 20일부터 한강에서 벚꽃 축제가 열렸습니다....</p>  <div itemprop="image" itemscope itemtype="https://schema.org/ImageObject">  <img itemprop="image" src="img/sakura.jpg" alt="벚꽃" />  <meta itemprop="url" content="http://exsample.com/img/sakura.jpg">  <meta itemprop="width" content="350">  <meta itemprop="height" content="200">  </div>  <p itemprop="author">윤인성</p>  <div itemprop="publisher" itemscope itemtype="http://schema.org/Organization">  <meta itemprop="name" content="공원사무실">  <div itemprop="logo" itemscope itemtype="https://schema.org/ImageObject">  <img src="http://exsample.com/img/logo.jpg" />  <meta itemprop="url" content="http://exsample.com/img/logo.jpg">  <meta itemprop="width" content="50">  <meta itemprop="height" content="50">  </div>  </div>  </div> |

**코드 4.21 Microdata의 itemref 속성**

|  |
| --- |
| <div itemscope itemtype="http://schema.org/Article" itemprop="mainEntity" itemref="article">  <p itemprop="headline">한강 벚꽃 축제</p>  <p itemprop="datePublished">2017/04/10</p>  <p itemprop="articleBody">2017년 04월 20일부터 한강에서 벚꽃 축제가 열렸습니다....</p>  </div>  <div id="article">  <p itemprop="author">윤인성</p>  <!-- 일부 필수 요소 생략 -->  </div> |

**코드 4.22 RDFa Lite의 vocab 속성, typeof 속성, property 속성**

|  |
| --- |
| <div vocab="http://schema.org/" typeof="Article">  <p property="headline">한강 벚꽃 축제</p>  <p property="datePublished">2017/04/10</p>  <p property="articleBody">2017년 04월 20일부터 한강에서 벚꽃 축제가 열렸습니다....</p>  <p property="author">윤인성</p>  <!-- 일부 필수 요소 생략 -->  </div> |

**코드 4.23 RDFa Lite의 resource 속성**

|  |
| --- |
| <p property="author" resource="#han.sakura" typeof="Person">  <span property="name">윤인성</ span>  </p> |

**코드 4.24 RDFa Lite의 prefix 속성**

|  |
| --- |
| <div vocab="http://schema.org/" prefix="foaf :http://xmlns.com/foaf/0.1/" typeof="Article">  <p property="headline">한강 벚꽃 축제</p>  <p property="datePublished">2017/04/10</p>  <p property="articleBody">2017년 04월 20일부터 한강에서 벚꽃 축제가 열렸습니다....</p>  <div typeof="foaf:Person">  <p property="foaf:name">윤인성</p>  <p property="foaf:birthday">11-11</p>  </div>  <!-- 일부 필수 요소 생략 -->  </div> |

**코드 4.25 JSON-LD**

|  |
| --- |
| <head>  <script type="application/ld+json"> ❶  {  "@context": "http://schema.org", ❷  "@type": "Article", ❸  "headline": "한강 벚꽃 축제",  "datePublished": "2017/04/10",  "articleBody": "2017년 04월 20일부터 한강에서 벚꽃 축제가 열렸습니다.",  "author": "윤인성",  "image":{  "@type": "ImageObject",  "height": "240px",  "width": "360px",  "author": "윤인성",  "contentLocation": "한강 공원",  "contentUrl": "sakura.jpg",  "url": "http://exsample.com/img/sakura.jpg",  "datePublished": "2017/04/10",  "description": "벚꽃 사진 모음",  "name": "벚꽃"  }  }  <!-- 일부 필수 요소 생략 -->  </script>  </head>  <body>  <div>  <p>한강 벚꽃 축제</p>  <!-- 생략 -->  </div>  </body> |

**코드 5.1 Authorization 헤더로 ID와 비밀번호 전송하기**

|  |
| --- |
| curl -H 'Authorization: Basic aWQ6cGFzc3dk' http://www.example.com/ |

**코드 5.2 ID와 비밀번호를 URL에 포함시켜 전송하기**

|  |
| --- |
| curl http://id:password@www.example.com/ |

**코드 5.3 Jsoup로 Basic 인증이 걸려 있는 웹 페이지에 요청하기**

|  |
| --- |
| package kr.co.rint.crawlerbook;  import org.jsoup.Connection.Response;  import org.jsoup.Connection.Method;  import org.jsoup.HttpStatusException;  import org.jsoup.Jsoup;  import org.jsoup.nodes.Document;  import java.io.IOException;  import java.util.Base64;  public class BasicAuthentication {  public static void main(String[] args) {  String url = "http://localhost:8080/";  String username = "username";  String password = "password";  // 사용자 이름과 비밀번호를 콜론(:)으로 연결한 뒤 Base64로 인코딩하기  String authorization = username + ":" + password;  String base64Authorization = new String(Base64.getEncoder().encodeToString(authorization.getBytes()));  try {  Response res = Jsoup.connect(url).method(Method.HEAD).execute();  // HTTP 인증이 필요하지 않은 페이지의 경우  // 따로 예외가 발생하지 않고 try 구문을 그대로 진행함  Document doc = Jsoup.connect(url).get();  ...  } catch(HttpStatusException e) {  // HTTP 인증이 필요한 페이지의 경우 예외가 발생하므로 catch 구문으로 들어옴  Integer status = e.getStatusCode();  // HTTP 인증이 필요한 페이지는  // 상태 코드 401(Authorization Required)를 응답함  if (status == 401) {  try {  // Authorization 헤더에 Base64 인코딩한 데이터를 넣어 요청하기  Document doc = Jsoup.connect("http://localhost:8080/")  .header("Authorization", "Basic " + base64Authorization)  .get();  ...  } catch (IOException e2) {  e2.printStackTrace();  }  }  } catch (IOException e) {  e.printStackTrace();  }  }  } |

**코드 5.4 GitHub 로그인 화면의 소스 코드**

|  |
| --- |
| <form accept-charset="UTF-8" action="/session" method="post">  <input name="utf8" type="hidden" value=" ✓ ">  <input name="authenticity\_token" type="hidden" value="XXXXXXXXXXXXXXXXXXXXXX==">  <h1>Sign in to GitHub</h1>  <label for="login\_field">Username or email address</label>  <input id="login\_field" name="login" type="text">    <label for="password">Password</label>  <input id="password" name="password" type="password">    <input name="commit" type="submit" value="Sign in">  </form> |

**코드 5.5 Jsoup를 사용해 GitHub에 로그인하기**

|  |
| --- |
| package kr.co.rint.crawlerbook;  import org.jsoup.Connection;  import org.jsoup.Connection.Response;  import org.jsoup.Connection.Method;  import org.jsoup.Jsoup;  import org.jsoup.nodes.Document;  import java.io.IOException;  import java.net.URLEncoder;  import java.util.HashMap;  import java.util.Map;  public class FormAuthentication {  public static void main(String[] args) {  try {  // ① 로그인 페이지의 HTML 추출하기  Response res = Jsoup.connect("https://github.com/login")  .userAgent("User-Agent:Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_10\_4) AppleWebKit/600.7.12 (KHTML, like Gecko) Version/8.0.7 Safari/600.7.1")  .header("Referer", "https://github.com/")  .header("Accept", "text/html,application/xhtml+xml, application/xml;q=0.9,image/webp,image/apng,\*/\*;q=0.8")  .header("Accept-Encoding", "gzip, deflate, br")  .header("Accept-Language", "en-US,en;q=0.8")  .header("Host", "github.com")  .header("Origin", "https://github.com")  .header("Connection", "keep-alive")  .header("Cache-Control", "max-age=0")  .header("Upgrade-Insecure-Requests", "1")  .method(Method.GET).execute();  Document doc = res.parse();  // ② 첫 응답의 쿠키를 추출해 저장해두기  Map<String, String> cookies = res.cookies();  // ③ CSRF를 피하기 위한 토큰 값 추출하기  final String authenticity\_token = doc.select("input[name=authenticity\_token]").first().attr("value");  // GitHub ID 입력하기  final String login = "my\_username";  // GitHub 비밀번호 입력하기  final String password = "my\_password";  // ④ 로그인 요청으로 전송할 값을 URL 인코딩하고  // 요청 바디의 길이 구하기  Map<String, String> formData = new HashMap();  formData.put("utf-8", " ✓ ");  formData.put("authenticity\_token", authenticity\_token);  formData.put("login", login);  formData.put("password", password);  formData.put("commit", "Sign in");  StringBuilder sb = new StringBuilder();  for(Map.Entry<String, String> entry: formData.entrySet()){  if(sb.length() > 0){  sb.append("&");  }  sb.append(entry.getKey() + "=" + URLEncoder.encode(entry.getValue(), "UTF-8").replace("%20", "+"));  }  String requestBody = new String(sb);  // 로그인 처리 요청 보내기  Connection con2 = Jsoup.connect("https://github.com/session")  // ⑤ 요청 헤더 설정하기  .userAgent("User-Agent:Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_10\_4) AppleWebKit/600.7.12 (KHTML, like Gecko) Version/8.0.7 Safari/600.7.1")  .header("Content-Type", "application/x-www-form-urlencoded")  .header("Referer", "https://github.com/")  .header("Accept", "text/html,application/xhtml+xml, application/xml;q=0.9,image/webp,image/apng,\*/\*;q=0.8")  .header("Accept-Encoding", "gzip, deflate, br")  .header("Accept-Language", "en-US,en;q=0.8")  .header("Host", "github.com")  .header("Origin", "https://github.com")  .header("Connection", "keep-alive")  .header("Cache-Control", "max-age=0")  .header("Upgrade-Insecure-Requests", "1")  .header("Content-Length", Integer.toString(requestBody.length()))  // 이전에 저장한 쿠키 사용하기  .cookies(cookies)  .requestBody(requestBody)  .method(Method.POST)  // 로그인 후 Referer로 설정한 URL로 리다이렉트하지 않게 설정하기  .followRedirects(false);  Response res2 = con2.execute();  // 로그인 후의 응답에서 쿠키를 추출해 저장해두기  Map<String, String> cookies2 = res2.cookies();  Response res3 = Jsoup.connect("https://github.com/settings/profile")  .userAgent("User-Agent:Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_10\_4) AppleWebKit/600.7.12 (KHTML, like Gecko) Version/8.0.7 Safari/600.7.1")  .header("Referer", "https://github.com/")  .header("Accept", "text/html,application/xhtml+xml, application/xml;q=0.9,image/webp,\*/\*;q=0.8")  .header("Accept-Encoding", "gzip, deflate, br")  .header("Accept-Language", "en-US,en;q=0.8")  .header("Host", "github.com")  .header("Connection", "keep-alive")  // ⑥ 로그인 후의 요청에 저장한 쿠키를 넣어 전송하기  // 이렇게 하면 서버가 로그인한 사용자로 인식함  .cookies(cookies2)  .method(Method.GET)  .execute();  // 로그인 후 페이지에 접근하면  // 상태 코드가 200으로 나오는 것을 확인할 수 있음  System.out.println(res3.statusCode());  // 로그인 후 프로필 페이지에 접근해  // 계정과 관련된 정보 추출하기  Document doc3 = res3.parse();  String myName = doc3.select("#user\_profile\_name").attr("value");  System.out.println(myName);  } catch(IOException e) {  e.printStackTrace();  }  }  } |

**코드 5.6 AWS SDK를 사용해 Amazon S3에 접근하기**

|  |
| --- |
| String accessKey = System.getenv("AWS\_ACCESS\_KEY");  String secretKey = System.getenv("AWS\_SECRET\_KEY");  AWSCredentials credentials = new BasicAWSCredentials(accessKey, secretKey);  AmazonS3 s3 = new AmazonS3Client(credentials);  ... |

**코드 5.8 GitHub의 REST API에 요청하기(요청 매개 변수)**

|  |
| --- |
| curl https://api.github.com/?access\_token=OAUTH-TOKEN |

**코드 5.9 접근 토큰을 요청 바디에 포함하는 방법**

|  |
| --- |
| curl -X POST -H "Content-Type: application/x-www-form-urlencoded" --data "access\_token=OAUTH-TOKEN" https://api.example.com/path |

**코드 5.10 pom.xml에 pac4j 의존 관계 추가하기**

|  |
| --- |
| <dependency>  <groupId>org.pac4j</groupId>  <artifactId>pac4j</artifactId>  <version>2.1.0</version>  </dependency>  <dependency>  <groupId>org.pac4j</groupId>  <artifactId>pac4j-oauth</artifactId>  <version>2.1.0</version>  </dependency> |

**코드 5.11 pac4j로 GitHubClient 생성하기**

|  |
| --- |
| GitHubClient client = new GitHubClient(clientId, secret);  client.setCallbackUrl("http://localhost:8080/callback");  client.setScope("repo, user"); |

**코드 5.12 pac4j로 인증 요구하기**

|  |
| --- |
| J2EContext context = new J2EContext(request, response);  client.redirect(context); |

**코드 5.13 pac4j로 사용자 정보와 접근 토큰 추출하기**

|  |
| --- |
| J2EContext context = new J2EContext(request, response);  OAuth20Credentials credentials = client.getCredentials(context);  GitHubProfile profile = client.getUserProfile(credentials, context);  // 사용자 정보 또는 API 호출 전용 접근 토큰을 추출합니다.  String userName = profile.getUsername();  String email = profile.getEmail();  String accessToken = profile.getAccessToken(); |

|  |
| --- |
| GET https://api.github.com/user?access\_token=OAUTH\_TOKEN |

|  |
| --- |
| Authorization: token OAUTH-TOKEN |

**코드 6.1 robots.txt 예**

|  |
| --- |
| User-agent: \*  Crawl-delay: 5  Disallow: /test/  Disallow: /help # disallows /help/index.html etc.  Allow: /help/faq.html  Sitemap: http://www.example.com/sitemap.xml |

**코드 6.2 User-agent의 예[robots.txt]**

|  |
| --- |
| User-agent: Googlebot  ...Googlebot의 접근 제한 설정...  User-agent: \*  ...모든 크롤러의 접근 제한 설정...  User-agent: Sample Crawler  ...Sample Crawler의 접근 제한 설정... |

|  |
| --- |
| User-agent: Sample Crawler  Disallow: / |

**코드 6.4 /help/ 디렉터리 내부에서 faq.html의 접근만 허가하기[robots.txt]**

|  |
| --- |
| User-agent: Sample Crawler  Disallow: /help/  Allow: /help/faq.html |

**코드 6.5 모든 접근 허가하기[robots.txt]**

|  |
| --- |
| User-agent: Sample Crawler  Disallow: |

**코드 6.6 /test 디렉터리 아래에 배치된 모든 페이지의 접근 금지하기**

|  |
| --- |
| User-agent: Sample Crawler  Disallow: /test/ |

**코드 6.7 /test로 시작하는 모든 디렉터리와 파일의 접근 금지하기[robots.txt]**

|  |
| --- |
| User-agent: Sample Crawler  Disallow: /test |

**코드 6.8 /readme 파일만 접근 금지하기[robots.txt]**

|  |
| --- |
| Disallow: /readme$ |

**코드 6.9 /test로 시작하는 디렉터리만 접근 금지하기[robots.txt]**

|  |
| --- |
| Disallow: /test\*/ |

**코드 6.10 모든 png 파일의 접근 금지하기[robots.txt]**

|  |
| --- |
| Disallow: /\*.png$ |

**코드 6.11 Disallow와 Allow는 구체적으로 경로를 지정할수록 우선순위가 높음[robots.txt]**

|  |
| --- |
| User-agent: Sample Crawler  Allow: /  Disallow: /test/  Disallow: /help |

**코드 6.12 Disallow와 Allow의 지정이 같은 경우[robots.txt]**

|  |
| --- |
| User-agent: \*  Disallow: /test/  Allow: /test/ |

**코드 6.A crawler-commons를 사용해 robots.txt 분석하기**

|  |
| --- |
| import crawlercommons.robots.BaseRobotRules;  import crawlercommons.robots.SimpleRobotRulesParser;  import java.util.List;  public class RobotParserSample {  public static void main(String[] args) {  // robots.txt를 읽어 들입니다.  byte[] content = ...  SimpleRobotRulesParser parser = new SimpleRobotRulesParser();  // 매개 변수는 순서대로  // 1. URL(로그 출력에 사용)  // 2. robots.txt의 내용  // 3. robots.txt의 Content-Type  // 4. 크롤러의 이름  BaseRobotRules rules = parser.parseContent("http://www.example.com", content, "text/plain", "Sample Crawler");  // 크롤링해도 괜찮은 경우에는 true를 리턴합니다.  boolean isAllowed = rules.isAllowed("http://www.example.com/help/faq.html");  // 사이트맵  List<String> sitemaps = rules.getSitemaps();  ...  }  } |

**코드 6.13 robots meta 태그의 예**

|  |
| --- |
| <html>  <head>  <meta name="robots" content="noindex" />  </head>  ...  </html> |

**코드 6.14 인덱스는 허가하지만 링크 순회는 허가하지 않는 robots meta 태그**

|  |
| --- |
| <meta name="robots" content="index, nofollow" /> |

**코드 6.15 nofollow가 링크에 설정돼 있는 경우**

|  |
| --- |
| <a href="http://www.example.com/index.html" rel="nofollow">샘플 사이트</a> |

**모든 크롤러에 인덱스 금지 및 링크 접근 금지하기**

|  |
| --- |
| HTTP/1.1 200 OK  Server: nginx  Date: Thu, 25 May 2017 20:35:41 GMT  X-Robots-Tag: none  ... |

**특정 크롤러만 대상으로 지정하기**

|  |
| --- |
| X-Robots-Tag: Googlebot: none  X-Robots-Tag: Sample Crawler: index, nofollow |

|  |
| --- |
| <head>  <link rel="prev" href="http://www.example.com/search.html?page=1">  <link rel="next" href="http://www.example.com/search.html?page=3">  </head> |

|  |
| --- |
| <a href="/search.html?page=1" class="active">1</a>  <a href="/search.html?page=2">2</a> |

|  |
| --- |
| a.active + a |

|  |
| --- |
| <ul>  <li class="prev"><a href="/search.html?page=2">이전</a></li>  <li><a href="/search.html?page=2">2</a></li>  <li class="active"><a href="/search.html?page=3">3</a></li>  <li class="next"><a href="/search.html?page=4">다음</a></li>  </ul> |

|  |
| --- |
| <ul>  <li class="prev"><a href="/search.html?page=9">이전</a></li>  <li><a href="/search.html?page=9">9</a></li>  <li class="active"><a href="/search.html?page=10">10</a></li>  <li class="next"><a href="/search.html?page=10">다음</a></li>  </ul> |

**코드 6.16 크롤러는 robots.txt의 Sitemap을 보고 사이트맵이 있는 곳을 찾습니다.**

|  |
| --- |
| User-agent: \*  Allow: /  Sitemap: http://www.example.com/sitemap.xml |

**코드 6.17 사이트맵 XML**

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">  <url>  <loc>http://www.example.com/</loc>  <lastmod>2017-07-03</lastmod>  <changefreq>weekly</changefreq>  <priority>0.6</priority>  </url>  </urlset> |

**코드 6.18 사이트맵 인덱스 파일**

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <sitemapindex xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">  <sitemap>  <loc>http://www.example.com/sitemap1.xml.gz</loc>  <lastmod>2017-07-03T07:30:00+09:00</lastmod>  </sitemap>  <sitemap>  <loc>http://www.example.com/sitemap2.xml.gz</loc>  <lastmod>2018-01-10</lastmod>  </sitemap>  </sitemapindex> |

**코드 6.19 사이트맵 읽어 들이기**

|  |
| --- |
| public void parse(InputStream stream, String charsetName) throws IOException {  BufferedInputStream bis = new BufferedInputStream(stream);  bis.mark(512);  // 앞의 512바이트를 읽어 들입니다.  byte[] bytes = new byte[512];  bis.read(bytes, 0, 512);  String start = new String(bytes, charsetName);  bis.reset();  // 사이트맵 XML  if (start.contains("<urlset")) {  ...  // 사이트맵 인덱스  } else if (start.contains("<sitemapindex")) {  ...  // 텍스트 파일로 된 사이트맵  } else if (start.matches("^https?://.\*"))  ...  // gzip 형식  } else {  GZIPInputStream gz = new GZIPInputStream(bis);  parse(gz, charsetName);  }  } |

**코드 6.A crawler-commons로 사이트맵 분석하기**

|  |
| --- |
| import crawlercommons.sitemaps.\*;  import java.io.IOException;  import java.net.URL;  public class SiteMapParserSample {  public static void main(String[] args) throws IOException, UnknownFormatException {  byte[] content = ...  URL sitemapUrl = new URL("http://www.example.com/sitemap.xml");  SiteMapParser parser = new SiteMapParser();  AbstractSiteMap sitemap = parser.parseSiteMap(content, sitemapUrl);  // 사이트맵 인덱스 파일인지 확인하기  if (sitemap.isIndex()) {  SiteMapIndex smIndex = (SiteMapIndex) sitemap;  for (AbstractSiteMap sm: smIndex.getSitemaps()) {  // (예) http://www.example.com/sitemap1.xml.gz  URL url = sm.getUrl();  ...  }  } else {  SiteMap sm = (SiteMap) sitemap;  for (SiteMapURL u: sm.getSiteMapUrls()) {  // (예) http://www.example.com/catalog?item=1  URL url = u.getUrl();  ...  }  }  }  } |

**코드 6.B 거대한 크기의 사이트 맵을 분석할 때 SiteMapParserSAX 사용하기**

|  |
| --- |
| SiteMapParser parser = new SiteMapParserSAX();  AbstractSiteMap sitemap = parser.parseSiteMap(content, sitemapUrl);  ... |

**코드 6.C crawler-commons로 Atom 다루기**

|  |
| --- |
| URL atomUrl = new URL("http://www.example.com/atom.xml");  SiteMapParser parser = new SiteMapParserSAX();  SiteMap atom = (SiteMap) parser.parseSiteMap(content, atomUrl);  int count = atom.getSiteMapUrls().size(); |

**코드 6.20 RSS 1.0의 예**

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8" ?>  <rdf:RDF xmlns="http://purl.org/rss/1.0/" xml:lang="ja"  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"  xmlns:dc="http://purl.org/dc/elements/1.1/">  <channel rdf:about="http://www.example.com/news.rss">  <!--웹 사이트의 타이틀(필수) -->  <title>example.com</title>  <!--웹 사이트의 URL(필수) -->  <link>http://www.example.com</link>  <!--웹 사이트의 설명(필수) -->  <description>This is an example.</description>  <!-- RSS 피드의 최종 변경일 -->  <dc:date>2017-07-03T07:30:00+09:00</dc:date>  <!-- 페이지 URL 목록 -->  <items>  <rdf:Seq>  <rdf:li rdf:resource="http://www.example.com/content.html" />  </rdf:Seq>  </items>  </channel>  <!-- 각 페이지의 정보 -->  <item rdf:about="http://www.example.com/content.html">  <title>Title</title>  <link>http://www.example.com/content.html</link>  <description>Description</description>  <dc:date>2017-07-02T23:10:20+09:00</dc:date>  </item>  </rdf:RDF> |

**코드 6.21 RSS 2.0의 예**

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8" ?>  <rss version="2.0">  <channel>  <!--웹 사이트의 타이틀(필수) -->  <title>example.com</title>  <!--웹 사이트의 URL(필수) -->  <link>http://www.example.com</link>  <!--웹 사이트의 설명(필수) -->  <description>This is an example.</description>  <!-- RSS 피드의 최종 변경일 -->  <lastBuildDate>Mon, 03 Jul 2017 07:30:00 +0900</lastBuildDate>  <!-- 페이지 정보 목록 -->  <item>  <title>Title</title>  <link>http://www.example.com/content.html</link>  <description>Description</description>  <pubDate>Sun, 02 Jul 2017 23:10:20 +0900</pubDate>  </item>  </channel>  </rss> |

**코드 6.22 Atom 1.0의 예**

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8" ?>  <feed xmlns="http://www.w3.org/2005/Atom" xml:lang="ja">  <!-- 피드의 유니크 식별자(필수) -->  <id>uuid:xxx</id>  <!--웹 사이트의 타이틀(필수) -->  <title>example.com</title>  <!-- 피드의 최종 변경일 -->  <updated>2017-07-03T07:30:00+09:00</updated>  <!--웹 사이트의 URL -->  <link href="http://www.example.com"/>  <!--웹 사이트의 설명 -->  <subtitle type="text">This is an example.</subtitle>  <!-- 페이지 정보 목록(id, title, updated는 필수) -->  <entry>  <id>uuid:xxx</id>  <title>Title</title>  <updated>2017-07-02T23:10:20+09:00</updated>  <link href="http://www.example.com/content.html"/>  <summary>Description</summary>  </entry>  </feed> |

**코드 6.23 유효 기간 확인하기**

|  |
| --- |
| $ curl -I https://tools.ietf.org/html/rfc7234  HTTP/1.1 200 OK  Date: Sun, 16 Apr 2017 16:29:36 GMT  Server: Apache/2.2.22 (Debian)  Content-Location: rfc7234.html  Vary: negotiate,Accept-Encoding  TCN: choice  Last-Modified: Sun, 09 Apr 2017 07:33:31 GMT  ETag: "225d6d7-1f3f3-54cb6e01848c0;54d4b2c8a73f2"  Accept-Ranges: bytes  Content-Length: 127987  Cache-Control: max-age=604800  Expires: Sun, 23 Apr 2017 16:29:36 GMT  Strict-Transport-Security: max-age=3600  X-Frame-Options: SAMEORIGIN  X-Xss-Protection: 1; mode=block  X-Content-Type-Options: nosniff  Content-Type: text/html; charset=UTF-8 |

.

**코드 6.24 Jsoup로 응답 헤더 참조하기**

|  |
| --- |
| String url = "https://tools.ietf.org/html/rfc7234";  Response res = Jsoup.connect(url).execute();  String expires = res.header("Expires");  // => Sun, 23 Apr 2017 16:29:36 GMT  String cacheControl = res.header("Cache-Control");  // => max-age=604800  String eTag = res.header("ETag");  // => "225d6d7-1f3f3-54cb6e01848c0;54d4b2c8a73f2"  String lastModified = res.header("Last-Modified");  // => Sun, 09 Apr 2017 07:33:31 GMT |

**코드 6.25 Jsoup로 요청 헤더를 지정해 접근하기**

|  |
| --- |
| Response res = Jsoup.connect(url)  .header("If-None-Match", eTag)  .header("If-Modified-Since", lastModified)  .execute();  int statusCode = res.statusCode(); |

**코드 2.26 클라이언트에서 "압축해서 전송해도 괜찮아요!"라고 언급하기**

|  |
| --- |
| $ curl -IL -H 'Accept-Encoding: gzip, deflate' http://en.wikipedia.org/ |

**코드 6.27 --compressed 옵션을 추가해서 요청하기**

|  |
| --- |
| $ curl -Lv --compressed http://en.wikipedia.org/ |

**코드 6.28 Jsoup를 사용하는 경우**

|  |
| --- |
| // 압축 전송된 경우에도 라이브러리가 자동으로 처리해줍니다.  Document doc = Jsoup.connect("http://en.wikipedia.org/").get();  Elements newsHeadlines = doc.select("#mp-itn b a"); |

**코드 6.29 Jsoup로 상태 코드 확인하기**

|  |
| --- |
| import org.jsoup.Connection.Response;  import org.jsoup.Jsoup;  import java.io.IOException;  public class NotFoundSample {  public void execute() throws IOException {  String url = "https://www.google.co.kr/123";  Response res = Jsoup.connect(url)  .ignoreHttpErrors(true)  .execute();  int statusCode = res.statusCode();  if (statusCode == 404) {  // ... 콘텐츠가 존재하지 않는 경우의 처리 ...  }  }  } |

**코드 6.30 Jsoup로 목록 페이지에서 링크 추출하기**

|  |
| --- |
| import org.jsoup.Jsoup;  import org.jsoup.nodes.Document;  import org.jsoup.nodes.Element;  import org.jsoup.select.Elements;  import java.io.IOException;  public class ListCrawlerSample {  public void start() throws Exception {  String url = "http://takezoe.hatenablog.com/";  // 목록 페이지만 크롤링하기(예제이므로 3페이지만 크롤링합니다)  for (int i = 0; i < 3; i++) {  Element nextUrl = execute(url);  if (nextUrl == null) {  break;  } else {  url = nextUrl.attr("href");  ...  }  }  // ... 출력되지 않은 것은 삭제된 것으로 간주하고 처리하기 ...  }  public Element execute(String url) throws IOException {  // GET 요청을 보내고 응답을 변수 doc에 저장하기  Document doc = Jsoup.connect(url).get();  // 추출한 HTML에서 링크 추출하기  Elements elements = doc.select("a.entry-title-link");  // 추출한 링크를 하나씩 처리  for (Element element: elements) {  // 링크 URL 추출(접근하지 않음)  String entryUrl = element.attr("href");  // ... entryUrl이 존재한다고 플래그를 남김 ...  }  // 다음 페이지의 링크 추출하기  return doc.select("a[rel=next]").first();  }  } |

**코드 7.1 이동 전에 확인하기**

|  |
| --- |
| <form method="/post" method="POST" onsubmit="return confirm('전송하겠습니까?');">  ...  <input type="submit" value="전송">  </form> |

**코드 7.2 링크 또는 버튼을 클릭했을 때 자바스크립트를 사용해 화면 이동하기**

|  |
| --- |
| <input type="text" id="keyword"/>  <input type="button" value="검색" onclick="search()"/>  <script>  function search(){  var keyword = document.getElementById('keyword').value;  location.href = 'http://example.com/search/' + encodeURIComponent(keyword);  }  </script> |

**코드 7.3 form 요소의 action 속성을 동적으로 변환하기**

|  |
| --- |
| <form method="POST" id="form">  ...  <input type="hidden" id="page" value="2"/>  <input type="submit" value="이전 페이지" onclick="prevPage()">  <input type="submit" value="다음 페이지" onclick="nextPage()">  </form>  <script>  /\*\*  \* [이전 페이지]를 클릭한 경우의 처리  \*/  function prevPage(){  var form = document.getElementById('form');  var page = parseInt(document.getElementById('page').value);  form.action = '/articles/' + (page - 1);  }  /\*\*  \* [다음 페이지]를 클릭한 경우의 처리  \*/  function nextPage(){  var form = document.getElementById('form');  var page = parseInt(document.getElementById('page').value);  form.action = '/articles/' + (page + 1);  }  </script> |

**코드 7.4 HTML을 동적으로 출력하기**

|  |
| --- |
| <body>  <script>  document.open();  document.write('<h1>자바스크립트로 출력하기</h1>');  document.close();  </script>  </body> |

**코드 7.5 HTML을 동적으로 변경하기**

|  |
| --- |
| <body>  <h1 id="title">HTML로 출력하기</h1>  <div id="content">HTML로 출력하기</div>  <script>  // 텍스트 변경하기  var h1 = document.getElementById('title');  h1.innerText = 'JavaScript로 출력하기';  // HTML 변경하기  var div = document.getElementById('content');  h1.innerHTML = '<b>JavaScript로 출력하기</b>';  </script>  </body> |

**코드 7.6 검색 결과 부분의 HTML을 자바스크립트로 생성하는 경우**

|  |
| --- |
| <form>  키워드: <input type="text" id="keyword"> <input type="button" id="button" value="검색"/>  </form>  <ul id="results">  <!-- 여기에 검색 결과를 출력합니다. -->  </ul>  <script>  var button = document.getElementById('button');  var keyword = document.getElementById('keyword');  button.onclick = function(){  // XMLHttpRequest 객체 생성하기  var xhr = new XMLHttpRequest();  // GET 요청 준비하기  xhr.open('GET', '/search?keyword=' + encodeURIComponent(keyword.value), true);  // XMLHttpRequest에 이벤트 핸들러 등록하기  xhr.onload = function(){  if (xhr.status === 200){  // 응답 내용을 JSON으로 파싱하기  var res = JSON.parse(xhr.responseText);  // JSON 데이터에 반복문을 적용한 뒤 HTML DOM 트리에 추가하기  var results = document.getElementById('results');  for(var i = 0; i < res.books.length; i++){  var li = document.createElement('li');  li.innerText = res.books[i].title;  results.appendChild(li);  }  }  };  xhr.send(null);  };  </script> |

**코드 7.7 POST 메서드 또는 PUT 메서드의 경우**

|  |
| --- |
| xhr.open('POST', '/search', true);  ...  xhr.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');  xhr.send('keyword=' + encodeURIComponent(keyword.value)); |

.

**코드 7.8 화면 이동을 자바스크립트로 구현한 페이지**

|  |
| --- |
| <form>  <input type="button" value="Next Page" onclick="goNextPage()"/>  <input type="hidden" id="page" value="10"/>  </form>  <script>  function goNextPage(){  // 현재 페이지 번호를 hidden 필드에서 추출하기  var page = parseInt(document.getElementById('page').value);  // "현재 페이지 번호 + 1"인 URL로 이동하기  location.href = 'http://example.com/items/list/' + (page + 1);  }  </script> |

**코드 7.9 자바스크립트 처리를 유사하게 모방하기**

|  |
| --- |
| Document doc = ...  // 현재 페이지를 hidden 필드에서 추출하기  int page = Integer.parseInt(doc.select("#id").val());  // "현재 페이지 번호 + 1"인 URL 생성하기  String nextUrl = "http://example.com/items/list/" + (page + 1);  // 다음 페이지에 요청 전송하기  Document nextDoc = Jsoup.connect(nextUrl).get(); |

**코드 7.10 페이징 전용 URL을 나타내는 link 태그**

|  |
| --- |
| <head>  <link rel="prev" href="http://www.example.com/search.html?page=1">  <link rel="next" href="http://www.example.com/search.html?page=3">  </head> |

|  |
| --- |
| <link rel="alternate" media="only screen and (max-width: 640px)" href="http://sp.example.com/page1" /> |

|  |
| --- |
| <link rel="canonical" href="http://www.example.com/page1"> |

**코드 7.11 pom.xml에 WebDriver 의존 관계 추가하기**

|  |
| --- |
| <dependency>  <groupId>org.seleniumhq.selenium</groupId>  <artifactId>selenium-java</artifactId>  <version>3.5.1</version>  </dependency> |

**코드 7.12 WebDriver로 크롤링하기**

|  |
| --- |
| package kr.co.rint.crawler;  import org.openqa.selenium.By;  import org.openqa.selenium.WebDriver;  import org.openqa.selenium.WebElement;  import org.openqa.selenium.phantomjs.PhantomJSDriver;  import org.openqa.selenium.support.ui.WebDriverWait;  import java.util.List;  public class WebDriverSample {  public static void main(String[] args) {  WebDriver driver = new PhantomJSDriver();  // Google 최상위 페이지에 접근하기  driver.get("http://www.google.com");  // q라는 텍스트 필드에 "WebDriver"라고 입력하고 요청 전송하기  WebElement element = driver.findElement(By.name("q"));  element.sendKeys("WebDriver");  element.submit();  // 타이틀이 "WebDriver"로 시작하는 문자열이 나올 때까지 대기하기  (new WebDriverWait(driver, 10)).until(d -> d.getTitle().startsWith("WebDriver"));  // 출력 결과에서 링크를 추출한 뒤 타이틀과 URL 출력하기  List<WebElement> elements = driver.findElements(By.cssSelector("h3.r>a"));  for(WebElement e: elements){  System.out.println(e.getText());  System.out.println(e.getAttribute("href"));  }  // 종료  driver.quit();  }  } |

**코드 7.13 WebDriver로 웹 페이지의 요소 추출하기**

|  |
| --- |
| WebElement element = driver.findElement(By.name("q")); |

**코드 7.14 WebDriver로 대화 상자 조작하기**

|  |
| --- |
| // 경고 대화 상자의 경우  Alert alertDialog = driver.switchTo().alert();  // 경고 대화 상자의 확인 버튼 클릭하기  alertDialog();  // 확인 대화 상자의 경우  Alert confirmDialog = driver.switchTo().alert();  // 확인 대화 상자의 확인 버튼 클릭하기  confirmDialog.accept();  // 확인 대화 상자의 취소 버튼 클릭하기  confirmDialog.dismiss();  // 입력 대화 상자의 경우  Alert inputDialog = driver.switchTo().alert();  // 입력 대화 상자에 문자열을 입력하고 확인 버튼 클릭하기  inputDialog.sendKeys("안녕하세요");  inputDialog.accept(); |

**코드 7.15 대화 상자에 출력된 글자 추출하기**

|  |
| --- |
| String text = alert.getText(); |

**코드 7.16 WebDriver로 버튼을 클릭하고 일정 시간 동안 대기하기**

|  |
| --- |
| // 검색 버튼 클릭하기  WebElement element = driver.findElement(By.id("search"));  element.click();  // 5초 대기하기  Thread.sleep(5000);  // 결과 추출하기  WebElement results = driver.findElement(By.id("results")); |

**코드 7.17 WebDriver로 암묵적으로 대기하기**

|  |
| --- |
| // 자동으로 대기할 시간을 10초로 설정하기  driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);  // 검색 버튼 클릭하기  WebElement element = driver.findElement(By.id("search"));  element.click();  // 명시적으로 대기하지 않아도 결과를 추출할 수 있음  WebElement results = driver.findElement(By.id("results")); |

**코드 7.18 WebDriver로 지정한 조건을 만족할 때까지 대기하기**

|  |
| --- |
| // 최대 10초까지 대기하기  WebDriverWait wait = new WebDriverWait(driver, 10);  // 검색 버튼 클릭하기  WebElement element = driver.findElement(By.id("search"));  element.click();  // id 속성이 "results"인 요소가 출력될 때까지 대기하기  ExpectedCondition<WebElement> condition = ExpectedConditions.presenceOfElementLocated(By.id("results"));  wait.until(condition); |

**코드 7.19 ExpectedConditions.not()으로 조건 반전하기**

|  |
| --- |
| // class 속성이 waiting이 아닐 때까지 대기하기  ExpectedCondition<Boolean> condition = ExpectedConditions.not(  ExpectedConditions.attributeToBe(By.id("result"), "class", "waiting")  );  wait.until(condition); |

**코드 7.20 ExpecteConditions.and()로 AND 조건 생성하기**

|  |
| --- |
| // 속성값이 success이며 텍스트가 Success일 때까지 대기하기  ExpectedCondition<Boolean> condition = ExpectedConditions.and(  ExpectedConditions.attributeToBe(By.id("result"), "class", "success"),  ExpectedConditions.textToBe(By.id("result"), "Success")  );  wait.until(condition); |

**코드 7.21 ExpectedConditions.or()로 OR 조건 생성하기**

|  |
| --- |
| // 속성값이 success 또는 info 중 하나가 될 때까지 대기하기  ExpectedCondition<Boolean> condition = ExpectedConditions.or(  ExpectedConditions.attributeToBe(By.id("result"), "class", "success"),  ExpectedConditions.attributeToBe(By.id("result"), "class", "info")  );  wait.until(condition); |

**코드 7.A 특정 요청 헤더와 바디를 포함한 POST 요청을 curl 명령어로 전송하고 HTTP 응답 헤더와 바디를 확인하는 방법**

|  |
| --- |
| curl https://pc-shop.com/search -i -XPOST -H "If-Modified-Since: Sat, 19 Aug 2017 00:00:00 GMT" -b "session=12345" -d "type=laptop&min\_price=100000&max\_price=150000&display\_size=14&condition=new&..." |