Web에서 File Download 기능 구현하기

1. HTML의 anchor tag에 있는 download 속성을 사용하는 방법
   - 예: 샘플파일 다운받기
     · 위의 예시에서 href 속성의 값이 '/'로 시작하지 않았기 때문에, "data/myfile.xlsx"의 값은 상대경로를 의미하게 되며, 현재 페이지가 있는 디렉터리를 기준으로 파일의 위치를 찾게 된다. 즉, 다운로드 링크를 클릭할 경우, 현재 페이지가 저장된 위치의 data 디렉터리 속에서 myfile.xlsx라는 파일을 찾아서 다운로드 하게 된다.
   - 예: 샘플파일 다운받기
     · download 속성은 이전의 예에서 처럼 값이 없이 사용될 수도 있지만, 위와 같이 값과 함께 사용될 경우에는 "sample.xlsx"라는 이름의 파일로 다운로드 받을 것을 권유하는 형식이 된다.
   - 예: 샘플파일 다운받기
     · 이 예에서는 href 속성의 값이 '/'로 시작하기 때문에, "/data/myfile.xlsx"의 값은 절대경로를 의미하게 된다. 즉, 웹 사이트의 루트(/)에 있는 data 디렉터리에서 myfile.xlsx라는 이름의 파일을 찾아서 다운로드 하게 된다.
   - Anchor의 download 속성을 사용할 때에는 파일의 이름이 한글이더라도 별도의 작업없이 브라우저에 의해서 파일 이름이 깨지는 현상이 없이 정상적으로 동작한다.

2. Servlet/JSP에서 File I/O를 통해 직접 다운로드 받는 방법
   - 다운로드할 파일의 FileInputStream을 열고, Response 객체의 OutputStream을 얻어서, FileInputStream으로부터 read(…)한 내용을 OutputStream에 write(…)하는 방식으로 직접 다운로드 처리를 할 수 있다. 이때, Response를 통해 웹 브라우저에 보내줄 Header에는 파일의 확장자를 참고해서 Content-Type으로 적절한 Mime Type을 설정해 주거나 임의의 이진 파일을 나타내는 "application/octet-stream"을 지정해 주어야 한다. 이때, 브라우저에 파일 다운로드를 지시하는 "Content-Disposition" Header도 함께 지정해 주어야 한다.

```
(이전 생략)
43  protected void doPost(HttpServletRequest request, HttpServletResponse ressponse)
44          throws ServletException, IOException {
45      ServletContext context = this.getServletContext();
46      String fileName = request.getParameter("file");
47      String filePath = context.getRealPath("/image/" + fileName);
48      System.out.println("file path = " + filePath);
49
50      File toDownload = new File(filePath);
51      if (toDownload.exists()) {
52          String mimeType = context.getMimeType(filePath);
53          System.out.println("mimeType = " + mimeType);
54
55          response.setContentType((mimeType != null) ? mimeType : "application/octet-stream");
56          response.setContentLength((int) toDownload.length());
57
58          String encFileName = URLEncoder.encode(toDownload.getName(), "UTF-8");
59          response.setHeader("Content-Disposition", "attachment; filename=\"" + encFileName + "\"");
60
61          try ( ServletOutputStream outS = response.getOutputStream();
62                  FileInputStream fin = new FileInputStream(toDownload)) {
63              byte[] buffer = new byte[1024];
64              int read = 0;
65              while ((read = fin.read(buffer)) != -1)
66                  outS.write(buffer, 0, read);
67
68              fin.close();
69              outS.flush();
70              outS.close();
71          }
72      } else {
```

```
73          response.setContentType("text/html;charset=UTF-8");
74          response.getWriter().println("file not exist!");
75      }
76  }
    (이후 생략)
```

- 코드를 통해 전체 과정을 따라가 보면 다음과 같다.
  1) 45~47번 행: ServletContext를 통해서 다운로드할 파일의 절대 경로를 얻는다.
     - ✓ ServletContext의 getRealPath("/image/" + fileName) 메소드 호출은 현재 웹 애플리케이션의 루트(예: pilotWeb/)로부터 "image/" 디렉터리 아래에서 fileName에 해당하는 이름의 파일에 대응되는 절대 경로를 반환해 준다.
  2) 50~51번 행: 파일의 절대 경로를 사용하여 java.io.File 객체를 생성하고, 그런 파일이 실제로 존재하는 지를 먼저 체크한 후에 다운로드 관련 처리를 진행한다.
     - ✓ File toDownload = new File(filePath);
     - ✓ if (toDownload.exists()) { … }
  3) 52~55번 행: response의 body에 다운받을 파일의 내용을 적어서 반환하게 되므로, Content Type의 형식을 다운받을 파일의 확장자를 참고하여 설정해 준다. 만약 파일의 확장자가 잘 알려진 Mime Type에 대응된다면 이것을 써도 좋고, 별도로 대응되는 Mime Type이 없는 경우라면 모든 이진 파일에 대응될 수 있는 "application/octet-stream" 값을 사용해도 좋다.
     - ✓ response.setContentType(getServletContext().getMimeType(filePath));
     - ✓ 혹은 response.setContentType("application/octet-stream");
  4) 56번 행: 다운로드할 파일의 크기를 Content-Length HTTP Header로 지정해 준다.
     - ✓ response.setContentLength((int) toDownload.length());
  5) 59번 행: 웹 브라우저에게 파일 다운로드를 지시하기 위한 Content-Disposition HTTP Header를 설정한다. 이때 다운받을 파일의 이름도 filename 속성으로 추천해 준다.
     - ✓ response.setHeader("Content-Disposition", "attachment; filename=\"xx.pdf\"");
  6) 58번 행: 만약 다운받을 파일의 이름에 ASCII 형식이 아닌, 한글과 같은 Unicode 문자가 포함되어 있을 경우에는 위의 예제와 같이 Content-Disposition 헤더의 filename 속성값을 지정할 때, URLEncoder를 사용하여 UTF-8 형식으로 인코딩해 줘야 한다. 아직까지 filename 속성값을 Non-ASCII 문자를 지원하기 위한 표준은 제정되어 있지 않기 때문에, 이러한 UTF-8 인코딩 방식은 웹 브라우저의 지원 여부에 의존하는 방식이다.
     - ✓ String encFileName = URLEncoder.encode(toDownload.getName(), "UTF-8");
  7) 63~67번 행: 파일의 내용을 읽어서, HTTP Response의 body에 출력해 준다. 이때 I/O 속도의 향상을 위해서 적절한 크기의 byte[]을 버퍼 공간으로 활용하는 것이 좋다. 출력 스트림을 얻을 때에는 response.getWriter()를 사용하면 텍스트 기반의 출력을 발생하게 되므로, response.getOutputStream()을 사용해서 바이트 기반의 출력을 얻어야 한다.
     - ✓ ServletOutputStream outS = response.getOutputStream();
     - ✓ FileInputStream fin = new FileInputStream(down);
     - ✓ byte[] buffer = new byte[1024];
     - ✓ int read = 0;
     - ✓ while ((read = fin.read(buffer)) != -1)
     - ✓ outS.write(buffer, 0, read);
  8) 68~70번 행: 전체 내용을 모두 출력하였으면, 해당 I/O Stream을 닫아 준다.
     - ✓ fin.close();
     - ✓ outS.flush();
     - ✓ outS.close();
- JSP를 이용한 방식도 기본적으로 Servlet을 이용하는 방식과 동일하다.
  - 주의할 점은 JSP의 내장 객체로 제공되는 out 객체가 텍스트 기반의 Writer 객체이므로, 이진 파일을 제대로 다운받기 위해서는 out을 사용하지 않고, 앞서의 Servlet 예제와 마찬가지로 response.getOutputStream() 호출을 통해 바이트 기반의 이진 출력 스트림을 얻어서 사용해야 한다.