**웹 앱 개발**

**\* file upload 방법의 이해**

html input의 type을 file로 하면 File을 업로드 할 수 있는 element가 만들어지고, 보내지는Content-Type은 application/x-www-form-rulencoded로 전송이 된다. 이렇게 하면 단순히 선택한 File 이름만 날라가게 되므로 multipart/form-data로 변경을 해주어야 한다.

**\* file upload의 확장자 검사 및 썸네일 노출**

html input의 accept를 통해 확장자를 제한할 수 있지만 지원 브라우저가 적다는 단점이 있다. 때문에 change 이벤트를 통해 확장자를 검사할 수 있다.

|  |
| --- |
| elImage.addEventListener(‘change’ function(evt) {  const image= evt.target.files;  if(valid(image)) {  ...  } else {  ...  }  }); |

다음으로 이미지를 업로드하기 위해 파일 선택 후 썸네일을 보여줄 수 있다. 아래와 같이 마찬가지로 event를 추가한 후 createObjectURL을 통해 썸네일 데이터를 구해 img의 src에 넣어준다.

|  |
| --- |
| elImage.addEventListener(‘change’ function(evt) {  const image= evt.target.files;  if(valid(image)) {  var sumbContainer= document.querySelector(‘’);  **sumbContainer.src= window.URL.createObjectURL(image);**  } else {  ...  }  }); |

**\* 로깅이란?**

로깅이란 정보를 제공하는 일련의 기록인 로그를 생성하도록 시스템을 작성하는 활동을 의미한다. 로그를 기록 함으로써 재현하기 힘든 버그에 대한 정보, 성능에 관한 통계 정보와 같이 유용한 정보를 알아낼 수 있다.

**\* slf4j 설정하기**

다양한 로그 라이브러리들(log4j, logback 등등)을 하나의 통일된 방식으로 사용할 수 있도록 제공하는 로깅 Facade이다. 로깅에 대한 추상 레이어를 제공하는 interface의 모음이다. 때문에 어플리케이션 초기 버전에서 slf4j를 구현한 로그 라이브러리를 사용 하다가 더 성능이 좋은 라이브러리가 나오면(slf4j를 구현한) 어플리케이션 변경 없이 구현체를 변경할 수 있다.

**\* 파일 업로드/다운로드 컨셉 설명**

파일과 같은 대용량 데이터를 클라이언트 서버간 전송할 땐 multipart/form-data를 사용한다. 이 데이터는 body를 지정된 boundary로 나누어 각각의 데이터를 표현하는 방식이다. 기본적으로 HttpServletRequest는 이러한 데이터 형식을 처리하는 메소드를 제공 안하고 오직 요청 메시지 body의 inputStream만을 제공하므로 사용자는 데이터를 적절히 파싱해서 사용해야 한다. 일반적으로 직접 구현하기 보다는 관련 라이브러리(commons-fileupload, commons-io 같은)를 사용한다.

스프링의 DispatcherServlet은 multipart/form-data 요청이 올 경우 MultipartResolver를 사용해야 하므로 MultipartResolver Bean을 설정 해주어야 한다.

**\* 파일 업로드 구현하기**

**\* 파일 다운로드 구현하기**