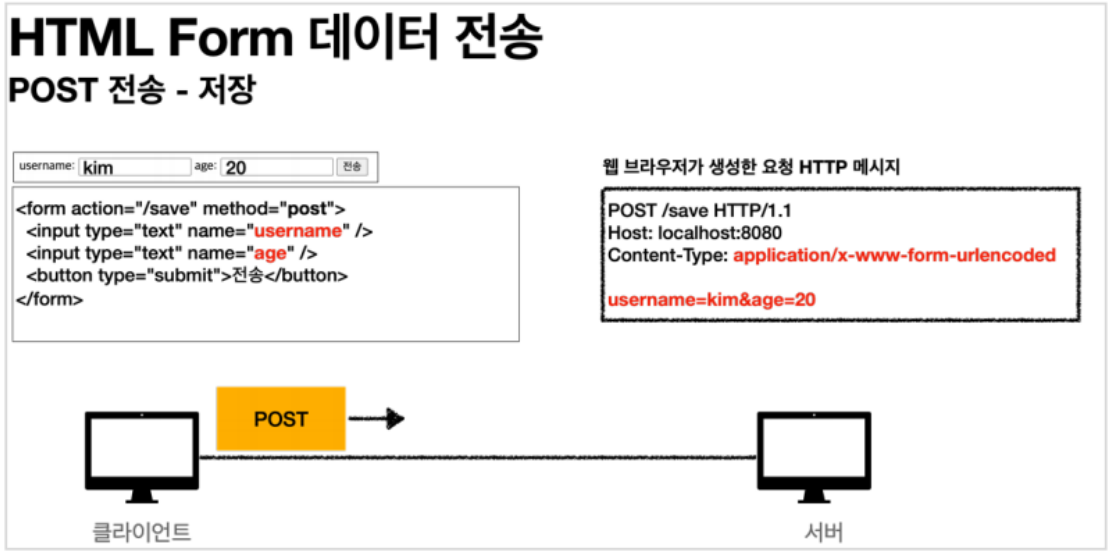
file upload

- `application/x-www-form-urlencoded`
- `multipart/form-data`

application/x-www-form-urlencoded 방식



- 파일을 업로드 하려면 파일은 문자가 아니라 바이너리 데이터를 전송해야 하

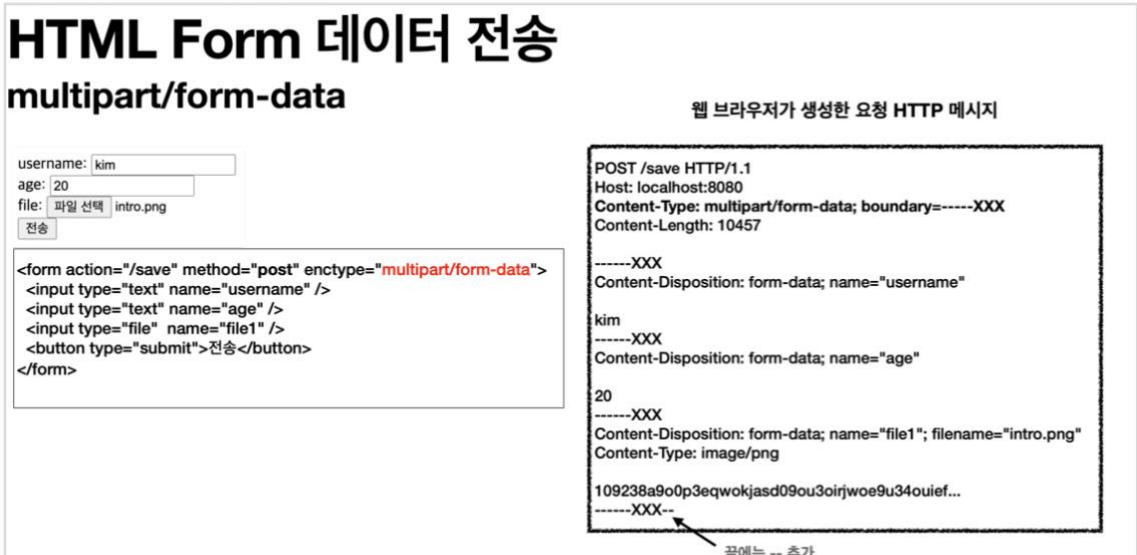
  폼을 이용해서 전송할 때 파일만 전송하지않는다.

- 이름
- 나이
- 첨부파일(프로필사진)

여기에서 이름과 나이도 전송해야 하고, 첨부파일도 함께 전송해야 한다. 문제는 이름과 나이는 문자로
전송하고, 첨부파일은 바이너리로 전송해야 한다는 점이다.
문자와 바이너리를 동시에 전송해야 하는 상황발생하.
이 문제를 해결하기 위해 **HTTP는 multipart/form-data 라는 전송 방식**을 제공한다.

---

multipart/form-data 방식



클라이언트 -> 서버 업로드 과정 이해하기

1. 파일업로드구현은 폼을 통해서 파일을 등록
2. Http메시지는 content-type 속성이 multipart/form-data 로 지정되고 정해진 형식에 따라 데이터를 인코딩하여 전송한다
3. 서버는 멀티파트 데이터에 대해서 각 파트별로 분리하여 개별적으로 정보를 얻는다

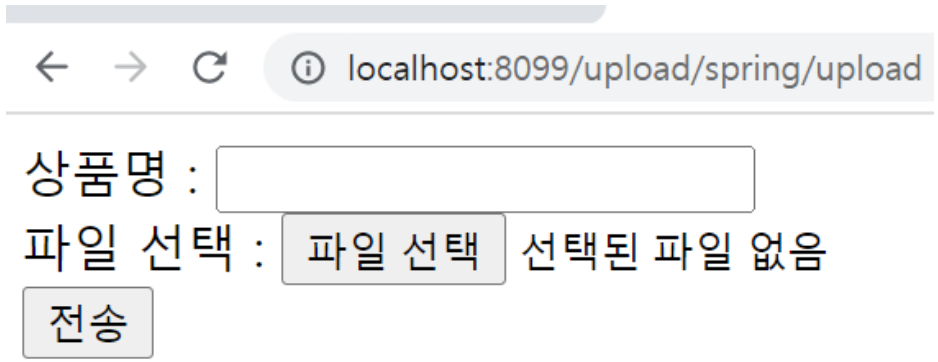
- entype 속성값

  - application/x-www-form-urlencoded (default 값)
  - multipart/form-data
    :파일전송시 반드시 multipart/form-data 로 지정 해야 함

- Multipart가 필요한 이유

HTTP Request는
Body에 클라이언트가 전송하려고 하는 데이터를 넣을 수 있다. (POST방식)
Body에 들어가는 데이터의 타입을 HTTP Header에 명시해 줌으로써 서버가 타입에 따라 알맞게 처리하게 한다.
이 Body의 타입을 명시하는 Header가 Content-type이다.
그런데 Content-type 타입을 하나만 명시할 수 있다. !!

한 Body에서 이 2 종류 이상의 데이터를 구분에서 넣어주는 방법도 필요해졌다.
그래서 multipart 타입이 필요해졌다.




```
<!-- 파일 업로드에서는 enctype(인코딩타입)을 multipart/form-data로 반드시 설정 -->
<form action="/upload/spring/upload" method="post" enctype="multipart/form-data">
    상품명 : <input type="text" name="itemName">
    <br>
   파일 선택 : <input type="file" name="file">
   <input type="submit" value="전송">
</form>
```

```
package com.acorn.upload;
 @Controller
@RequestMapping("/spring")
public class SpringUploadController {

    private String fileDir ="c:₩₩test₩₩upload₩₩";

    @GetMapping("/upload")
    public String newFile() {
        return "upload-form";
    }

    @PostMapping("/upload")
    public String saveFile( @RequestParam  String itemName,
                            @RequestParam MultipartFile file, Model model ) throws IOException {

        if (!file.isEmpty()) {
            String fullPath = fileDir + file.getOriginalFilename();
            file.transferTo(new File(fullPath));
            model.addAttribute("fileName" , file.getOriginalFilename());
        }
        return "upload-ok";
    }
     @ResponseBody
     @RequestMapping( value="/images/{fileName:.*}" , method=RequestMethod.GET)
     public Resource imageDownload(@PathVariable String  fileName) throws MalformedURLException {
                               System.out.println( "fileName"  + fileName);
              return new  UrlResource("file:c:₩₩test₩₩upload₩₩"+  fileName);
      }
}
```

파일업로드를 위한 dependency

pom.xml

```
<dependency>
    <groupId>commons-fileupload</groupId>
    <artifactId>commons-fileupload</artifactId>
    <version>1.3.1</version>
</dependency>

<!-- commons-io -->
<dependency>
    <groupId>commons-io</groupId>
    <artifactId>commons-io</artifactId>
    <version>2.4</version>
</dependency>
```

servlet-context.xml

```
<beans:bean id="multipartResolver" class="org.springframework.web.multipart.commons.CommonsMultipartResolver">

    <beans:property name="maxUploadSize" value="10485760" />
    <beans:property name="defaultEncoding" value="utf-8" />

</beans:bean>
```

1024*1024*10 => 10M