melius

melius102

.js

ode.js] FormData 객체 전송

;102 2020. 1. 29. 11:23

ext 데이터 전송

으로 form 태그를 이용하여 Text 데이터를 전송하는 방식은 아래와 같다.

라이언트 코드(with form tag)

```
- enctype="application/x-www-form-urlencoded" -->
rm id="form1" method="POST" action="/upload1">
  <input type="text" name="text1" value="text01">
  <input type="text" name="text2" value="text02">
  <button id="btn1">submit</button>
orm>
```

개발자 도구의 Network 탭에서 Request Headers를 확인하면,

tent-Type: application/x-www-form-urlencoded

m Data: text1=text01&text2=text02

통신으로 폼 데이터를 전송하기 위한 코드는 아래와 같다.

```
cript>
  btn1.onclick = function (e) {
      var formData = $("#form1").serialize();
      $.ajax({
          url: "/upload1",
          type: 'POST',
          data: formData,
          success: function (data) {
              alert(data);
          },
          error: function (xhr, status) {
              alert(xhr + " : " + status);
          }
      });
      return false;
  }
cript>
```

false 반환값은 버튼의 submit 기능을 차단하기 위한 코드이다.

라이언트 코드(with JSON)

을 이용하여 간단한 Text 데이터를 보낼수 있다.

```
 data = {
  text1: 'text01',
  text2: 'text02'

 xhr = new XMLHttpRequest();
.open("POST", "/upload1", false);
.setRequestHeader("Content-type", "application/json");
.send(JSON.stringify(data));
```

melius

melius102

버 코드

방식으로 전송된 form 데이터를 req 객체의 body 속성에 넣어주기 위해서 Node.js 서버에 아래와 같은 Body Parse
를 추가한다.

```
).use(express.urlencoded({ extended: false }));
).use(express.json());
```

에서 받은 req 객체의 body 속성에서 form 데이터의 값을 확인할 수 있다.

```
).post('/upload1', (req, res) => {
  console.log(req.headers);
  console.log(req.body.text1, req.body.text2);
  res.send("hello world");
```

Binary 데이터 전송

데이터가 아닌, Image 파일이나 Audio 파일과 같은 Binary 데이터를 서버로 전송하기 위해서는 form 태그의 enctyp
을 'multipart/form-data'로 설정한다.

라이언트 코드(with form tag)

```
orm id="form2" method="POST" action="/upload2" enctype='multipart/form-data'>
  <input type="text" name="text1" value="text01">
  <input type="file" id="files" name="file2" />
  <button id="btn2">submit</button>
orm>
```

개발자 도구의 Network 탭에서 Request Headers를 확인하면,
ntent-Type: multipart/form-data; boundary=----xxxxx....

통신으로 폼 데이터를 전송하기 위한 코드는 아래와 같다. Binary 데이터를 서버로 전송하기 위해서는 FormData 객체
하여 전달한다. Ajax 통신으로 전송시에는 form 태그의 enctype 속성을 생략해도 된다.

```
cript>
  btn2.onclick = function (e) {
      e.preventDefault();
      var form = $("#form2")[0];
      var formData = new FormData(form);

      $.ajax({
          url: "/upload2",
          // cache: false,
          processData: false,
          contentType: false,
          type: 'POST',
          data: formData,
          success: function (data) {
              alert(data);
          },
          error: function (xhr, status) {
              alert(xhr + " : " + status);
          }
      });
  }
cript>
```

melius

melius102

라이언트 코드(without form tag)

태그를 사용하지 않고 FormData 객체를 이용하여 직접 보낼 수도 있다.

```
iput type="file" id="file3" name="file3" />
itton id="btn3">submit</button>
:ript>
  btn3.onclick = function (e) {
      let formdata = new FormData();
      formdata.append("text3", "text03"); // text data
      formdata.append("file3", file3.files[0]); // binary data

      // use XMLHttpRequest
      // let xhr = new XMLHttpRequest();
      // xhr.open("POST", "/upload2", false);
      // // xhr.setRequestHeader("Content-type", "multipart/form-data");
      // xhr.send(formdata);

      // use jQuery
      $.ajax({
          url: '/upload2',
          type: 'POST',
          //async: true,
          //cache: false,
          contentType: false,
          processData: false,
          data: formdata,
          success: function (data) {
              alert(data);
          }
      });
  }
:cript>
```

버 코드

로 들어오는 Binary 데이터는 req 객체에서 확인이 가능하다.

```
).post('/upload2', (req, res) => {
  var contentType = req.headers['content-type'];
  console.log('contentType: ', contentType);
  var buffer = "";
  req.on('data', function (chunk) {
      buffer += chunk.toString();
  });

  req.on('end', function () {
      console.log('buffer: ', buffer);
      res.end('hello world');
  });
```

r 출력에서 확인하게 되면 전달된 데이터의 구조가 복잡하므로 개발자가 직접 분석하기는 쉽지 않으나. multer라는 오
을 사용하면 쉽게 데이터를 받을 수 있다.

```
ipm i multer
```

er 모듈을 이용한 서버단 코드는 아래와 같다. 우선 파일(Binary 데이터)이 저장될 폴더와 파일명을 지정한다. 아래 코
폴더의 이름을 'uploads'라고 지정하였고, getFile 함수는 전달된 파일 객체를 받아 서버에 저장될 새로운 이름을 생성
니다.

```
: storage = multer.diskStorage({
  destination: function (req, file, cb) {
      cb(null, 'uploads');
  },
  filename: function (req, file, cb) {
      cb(null, getFile(file));
  }
```

melius

melius102

```
ıction getFile(file) {
  let oriFile = file.originalname;
  let ext = path.extname(oriFile);
  let name = path.basename(oriFile, ext);
  let rnd = Math.floor(Math.random() * 90) + 10; // 10 ~ 99
  return Date.now() + '-' + rnd + '-' + name + ext;
```

ll 생성된 저장정보를 이용하여 아래와 같이 라우터 부분을 완성한다.

```
: upload = multer({
  storage: storage

 upload.none(): only for text-only multipart form
 upload.single('field-name'): only one file
).post(['/upload2', '/upload3'], upload.any(), (req, res) => {
  console.log(req.body);
  console.log(req.files);
  res.send("hello world");
```

1 구독하기

le.js' 카테고리의 다른 글

le.js] Worker Threads (0) 2020.0
le.js] FormData 객체 전송 (0) 2020.0
le.js] HTTPS 로컬 서버 구축 (0) 2020.0
le.js] TCP/IP 통신 (0) 2020.0

de.js' Related Articles

[Node.js] Worker Threads 2020.02.04

[Node.js] HTTPS 로컬 서버 구축 2020.01.29

[Node.js] TCP/IP 통신 2020.01.16

omments

Name Password

여러분의 소중한 댓글을 입력해주세요

Secret Send

Prev 1 ··· 31 32 33 34 35 36 37 38 39 ··· 76 Next

melius

melius102