

eXbuilder6

# 프로토콜 명세서



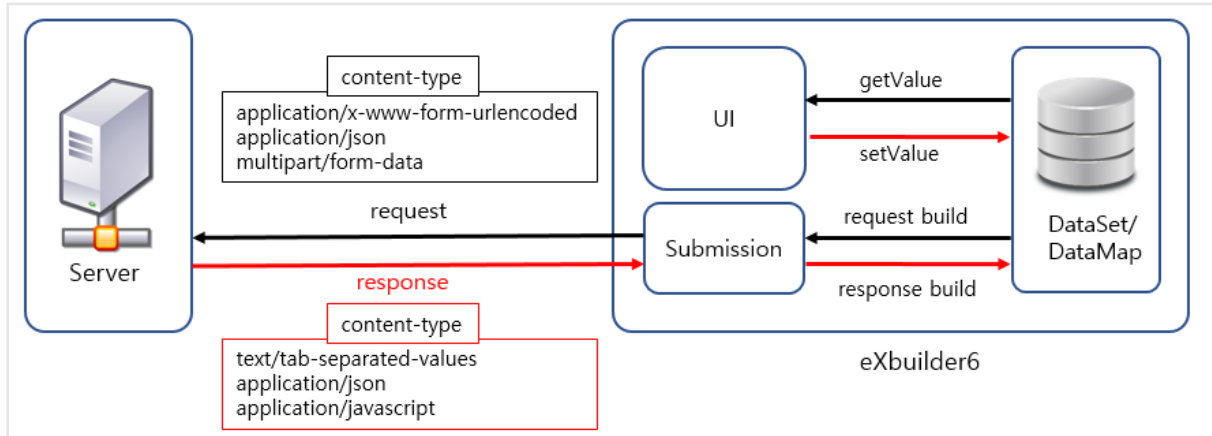
TOMATO SYSTEM

## 목 차

1.	개요 .....	3
2.	Submission.....	4
2.1.	request .....	6
2.1.1.	application/x-www-form-urlencoded.....	6
2.1.2.	application/json.....	8
2.1.3.	multipart/form-data .....	10
2.2.	response .....	13
2.2.1.	text .....	13
2.2.2.	javascript.....	16
2.2.3.	blob.....	17
2.2.4.	filedownload .....	18

## 1. 개요

서브미션의 request 하는 데이터 형식과 서브미션에서 response 로 받은 데이터 형식을 설명합니다. eXbuilder6 의 서브미션에서는 데이터셋과 데이터맵을 연결이 가능하며, response 로 받은 데이터를 데이터셋과 데이터맵에 자동으로 값을 지정해주는 기능을 갖고 있습니다. 이러한 기능으로 서브미션에 response 로 데이터를 전달할 때 eXbuilder6 에서 명세하는 데이터 형식을 사용해야 합니다.

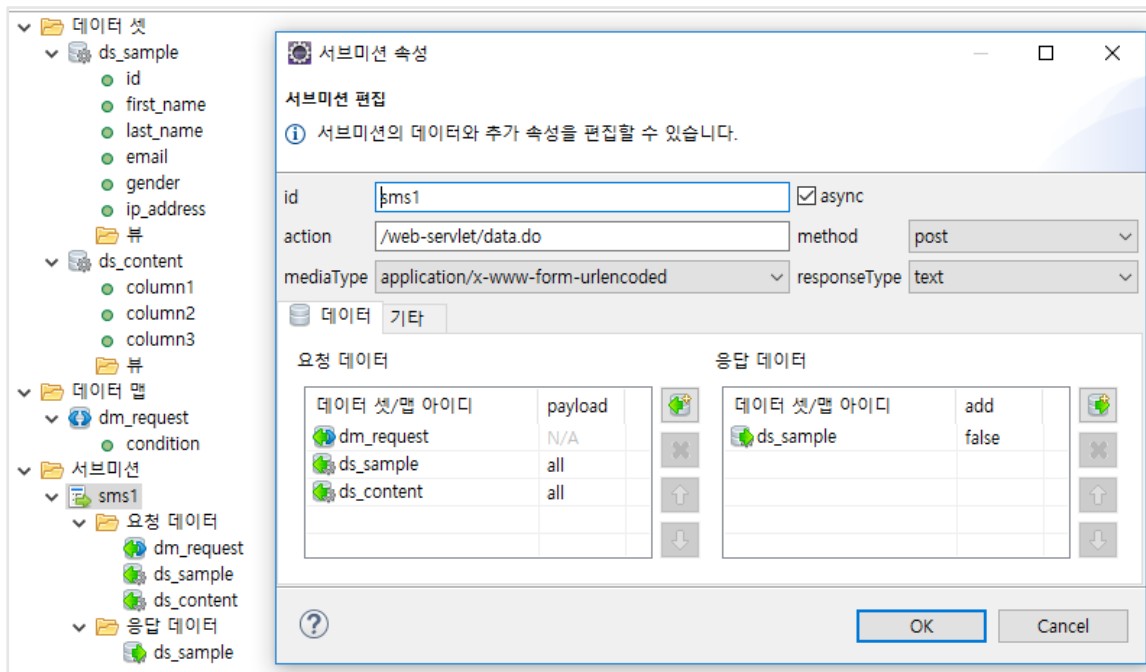


데이터 흐름

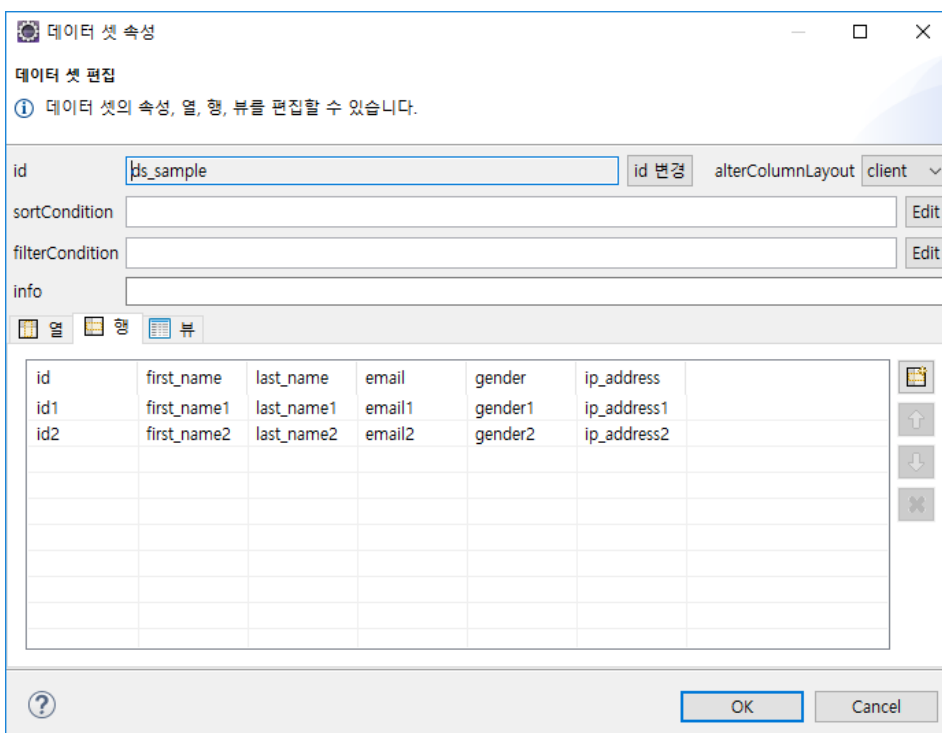
eXbuilder6 의 데이터는 [데이터 흐름]그림과 같이 request 와 response 에서 다양한 content-type 으로 데이터를 송수신하고 있습니다. 송수신 과정 중에 eXbuilder6 의 request 에는 데이터 전송 양식이 있으며, response 에는 TSV 와 JSON 에 있으니 확인하시기 바랍니다.

## 2. Submission

request 와 response 의 전송 규칙 및 설정에 따른 기능을 설명합니다. 다음 그림들은 request 와 response 에서 실제 네트워크에서 전송되는 내용의 서브미션의 설정입니다. 각각의 챔터에서 서브미션의 설정에 따른 데이터 전송을 확인 할 수 있습니다.



서브미션에 요청데이터, 응답데이터 설정



전송할 데이터셋의 값들

서브미션 속성

서브미션 편집

① 서브미션의 데이터와 추가 속성을 편집할 수 있습니다.

id  ☒ async

action  method

mediaType  responseType

데이터 기타

requestHeaders		requestParameters	
이름	값	이름	값
		param0	12
		param1	32

① OK Cancel

서브미션으로 전송할 파라미터 값들

## 2.1. request

타입	설명
<b>application/x-www-form-urlencoded</b>	기본 전송 방식입니다.
<b>application/json</b>	JSON 형태로 텍스트를 전송합니다.
<b>multipart/form-data</b>	파일을 포함하여 전송 할 수 있으며, post 방식만 지원합니다.

### 2.1.1. application/x-www-form-urlencoded

폼에서 전송하는 기본적인 방식이며 eXbuilder6 에서 정의된 형식으로 데이터를 보냅니다.

	내용
<b>지원 method</b>	GET, POST
<b>Request Header Content-Type</b>	application/x-www-form-urlencoded
<b>송신 데이터</b>	@d1#condition=&@d#=@d1#&@d1#=dm_request&

#### 2.1.1.1. 데이터 스키마

‘@d#’는 예약어입니다. 데이터셋과 데이터맵이 서브미션에 지정된 개수 만큼 숫자가 증가합니다.

데이터의 스키마 정의 파라미터: @d#

예) @d#=@d1#&@d#=@d2#

데이터셋 또는 데이터맵의 ID 정의 파라미터: @d1#, @d2#, ...

예) @d1#=ds\_sample&@d2#=dm\_request

데이터셋의 컬럼 정의 파라미터: @d1#이름

예) @d2#condition=조건검색&@d1#id=user

#### 2.1.1.2. 파싱 방법

‘@d#’의 파라미터 값에서 몇 개의 데이터셋과 데이터맵이 있는지 확인합니다.

‘@d1#’의 파라미터 값을 저장한 후 ‘@d1#condition’, ‘@d1#id’ 파라미터의 배열 순서대로 가져와서 Map 과 List 를 이용하여 저장합니다.

## 2.1.1.3. 실제 네트워크 결과

Name	×	Headers	Preview	Response	Cookies	Timing
sample1...		▼ General				
cleopatr...		Request URL: http://localhost:8080/web-servlet/data.do				
defaults.js		Request Method: POST				
languag...		Status Code: 200 OK				
user-mo...		Remote Address: [::1]:8080				
udc.js		Referrer Policy: no-referrer-when-downgrade				
echarts....		▼ Response Headers view source				
d3.min.js		Content-Type: application/json				
sample1...		Date: Thu, 08 Feb 2018 07:23:04 GMT				
checkboxo...		Server: Apache-Coyote/1.1				
data.do		Transfer-Encoding: chunked				
		▼ Request Headers view source				
		Accept: */*				
		Accept-Encoding: gzip, deflate, br				
		Accept-Language: ko,en-US;q=0.9,en;q=0.8,ja;q=0.7				
		Connection: keep-alive				
		Content-Length: 831				
		<u>Content-Type: application/x-www-form-urlencoded; charset=UTF-8</u>				
		Cookie: _ga=GA1.1.2133655032.1507789993				
		Host: localhost:8080				
		Origin: http://localhost:8080				
		Referer: http://localhost:8080/web-servlet/ui/sample/submission/sample1.clx				
		User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/64.0.3282.140 Safari/537.36				
		X-Requested-With: XMLHttpRequest				
		▼ Form Data view parsed				
		param0=12&param1=32&40d1%23condition=true&40d23=%40d1%23&40d1%23=dm_request&40d2%23id=id1&40d2%23first_name=first_name1&40d2%23last_name=last_name1&40d2%23email=email11&40d2%23gender=gender1&40d2%23ip_address=ip_address1&40d2%23sts=&40d2%23id=id2&40d2%23first_name=first_name2&40d2%23last_name=last_name2&40d2%23email=email12&40d2%23gender=gender2&40d2%23ip_address=ip_address2&40d2%23sts=&40d2%23=40d2%23&40d2%23=ds_sample&40d3%23column1=column11&40d3%23column2=column21&40d3%23column3=column31&40d3%23sts=&40d3%23column1=column12&40d3%23column2=column22&40d3%23column3=column32&40d3%23sts=&40d3%23column1=column13&40d3%23column2=column23&40d3%23column3=column33&40d3%23sts=&40d3%23column1=column14&40d3%23column2=column24&40d3%23column3=column34&40d3%23sts=&40d3%23=40d3%23&40d3%23=ds_content&				
11 requests ...						

## 2.1.1.4. 실제 네트워크의 request data

```
param0=12&param1=32&40d1%23condition=true&40d23=%40d1%23&40d1%23=dm_request&40d2%23id=id1&40d2%23first_name=first_name1&40d2%23last_name=last_name1&40d2%23email=email11&40d2%23gender=gender1&40d2%23ip_address=ip_address1&40d2%23sts=&40d2%23id=id2&40d2%23first_name=first_name2&40d2%23last_name=last_name2&40d2%23email=email12&40d2%23gender=gender2&40d2%23ip_address=ip_address2&40d2%23sts=&40d2%23=40d2%23&40d2%23=ds_sample&40d3%23column1=column11&40d3%23column2=column21&40d3%23column3=column31&40d3%23sts=&40d3%23column1=column12&40d3%23column2=column22&40d3%23column3=column32&40d3%23sts=&40d3%23column1=column13&40d3%23column2=column23&40d3%23column3=column33&40d3%23sts=&40d3%23column1=column14&40d3%23column2=column24&40d3%23column3=column34&40d3%23sts=&40d3%23=40d3%23&40d3%23=ds_content&
```

### 2.1.2. application/json

JSON 형식의 텍스트를 전송하기 위한 프로토콜입니다.

	내용
지원 method	GET, POST
Request Header Content-Type	application/json
송신 데이터	<pre>{   "param": {     "pa1": ["hi", "hello"],     "pa2": ["good"]   },   "data": {     "dm_request": {       "condition": "true",       "ds1": [         {           "id": "id1",           "first_name": "first_name1",           "last_name": "last_name1",           "email": "email1",           "gender": "gender1",           "ip_address": "ip_address1",           "sts": ""         },         {           "id": "id2",           "first_name": "first_name2",           "last_name": "last_name2",           "email": "email2",           "gender": "gender2",           "ip_address": "ip_address2",           "sts": ""         }       ]     }   } }</pre>

#### 2.1.2.1. 데이터 스키마

JSON에는 데이터컬렉션의 “data”키와 파라미터의 “param”키가 있습니다.

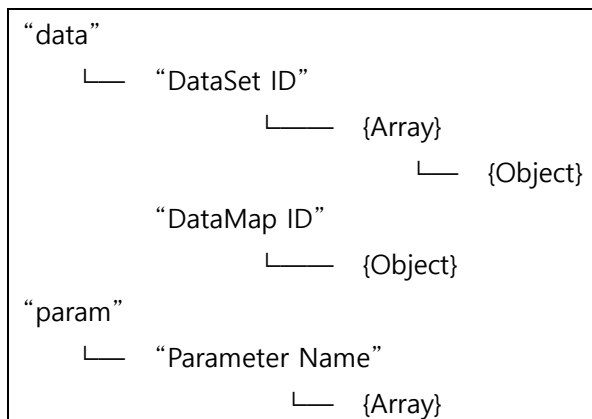
data: 데이터셋과 데이터맵의 내용이 포함됩니다.

예) {“data”: {“ds\_sample”: [{“id”: “user”, “name”: “kim”}, {“id”: “admin”, “name”: “park”}],  
“dm\_sample”: {“condition”: “ki”} } }

param: 파라미터의 내용이 포함됩니다.

예) {“param”: {“pa1”: [“hi”, “hello”], “pa2”: [“good”]} }

#### 2.1.2.2. 데이터 계층도





## 2.1.2.3. 실제 네트워크 결과

Name	×	Headers	Preview	Response	Cookies	Timing
sample1...		▶ General				
cleopatr...		▼ Response Headers	view source			
defaults.js		Connection: close				
languag...		Content-Language: en				
user-mo...		Content-Length: 1555				
udc.js		Content-Type: text/html; charset=utf-8				
echarts....		Date: Thu, 08 Feb 2018 07:44:21 GMT				
d3.min.js		Server: Apache-Coyote/1.1				
sample1...		▼ Request Headers	view source			
cleopatr...		Accept: */*				
cleopatr...		Accept-Encoding: gzip, deflate, br				
checkboxbo...		Accept-Language: ko,en-US;q=0.9,en;q=0.8,ja;q=0.7				
data.do		Connection: keep-alive				
		Content-Length: 685				
		<u>Content-Type: application/json; charset=UTF-8</u>				
		Cookie: _ga=GA1.1.2133655032.1507789993				
		Host: localhost:8080				
		Origin: http://localhost:8080				
		Referer: http://localhost:8080/web-servlet/ui/sample/submission/sample1.clx				
		User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/64.0.3282.140 Safari/537.36				
		X-Requested-With: XMLHttpRequest				
		▼ Request Payload	view parsed			
		<pre>{   "param": {     "param0": ["12"],     "param1": ["32"]   },   "data": {     "dm_request": {       "condition": "true"     },     "ds_sample": [       {         "id": "id1",         "first_name": "first_name1",         "last_name": "last_name1",         "email": "email1",         "gender": "gender1",         "ip_address": "ip_address1",         "sts": ""       },       {         "id": "id2",         "first_name": "first_name2",         "last_name": "last_name2",         "email": "email2",         "gender": "gender2",         "ip_address": "ip_address2",         "sts": ""       }     ],     "ds_content": [       {         "column1": "column11",         "column2": "column21",         "column3": "column31",         "sts": ""       },       {         "column1": "column12",         "column2": "column22",         "column3": "column32",         "sts": ""       },       {         "column1": "column13",         "column2": "column23",         "column3": "column33",         "sts": ""       },       {         "column1": "column14",         "column2": "column24",         "column3": "column34",         "sts": ""       }     ]   } }</pre>				

## 2.1.2.4. 실제 네트워크의 request data

```
{
  "param": {
    "param0": ["12"],
    "param1": ["32"]
  },
  "data": {
    "dm_request": {
      "condition": "true"
    },
    "ds_sample": [
      {
        "id": "id1",
        "first_name": "first_name1",
        "last_name": "last_name1",
        "email": "email1",
        "gender": "gender1",
        "ip_address": "ip_address1",
        "sts": ""
      },
      {
        "id": "id2",
        "first_name": "first_name2",
        "last_name": "last_name2",
        "email": "email2",
        "gender": "gender2",
        "ip_address": "ip_address2",
        "sts": ""
      }
    ],
    "ds_content": [
      {
        "column1": "column11",
        "column2": "column21",
        "column3": "column31",
        "sts": ""
      },
      {
        "column1": "column12",
        "column2": "column22",
        "column3": "column32",
        "sts": ""
      },
      {
        "column1": "column13",
        "column2": "column23",
        "column3": "column33",
        "sts": ""
      },
      {
        "column1": "column14",
        "column2": "column24",
        "column3": "column34",
        "sts": ""
      }
    ]
  }
}
```

### 2.1.3. multipart/form-data

파일과 정보를 같이 보내기 위한 프로토콜입니다.

	내용
지원 method	POST
Request Header Content-Type	multipart/form-data
송신 데이터	<pre> -----WebKitFormBoundarygDrd8VhjeByqso6r Content-Disposition: form-data; name="@d1#condition"  true -----WebKitFormBoundarygDrd8VhjeByqso6r Content-Disposition: form-data; name="@d#"  @d1# -----WebKitFormBoundarygDrd8VhjeByqso6r Content-Disposition: form-data; name="@d1#"  dm_request -----WebKitFormBoundarygDrd8VhjeByqso6r Content-Disposition: form-data; name="@d2#id" </pre>

#### 2.1.3.1. 데이터 스키마

‘@d#’는 예약어입니다. 데이터셋과 데이터맵이 서브미션에 지정된 개수 만큼 숫자가 증가합니다.

데이터의 스키마 정의 파라미터: @d#

예) @d#=@d1#&@d#=@d2#

데이터셋 또는 데이터맵의 ID 정의 파라미터: @d1#, @d2#, ...

예) @d1#=ds\_sample&@d2#=dm\_request

데이터셋의 컬럼 정의 파라미터: @d1#이름

예) @d2#condition=조건검색&@d1#id=user

#### 2.1.3.2. 파싱 방법

‘@d#’의 파라미터 값에서 몇 개의 데이터셋과 데이터맵이 있는지 확인합니다.

‘@d1#’의 파라미터 값을 저장한 후 ‘@d1#condition’, ‘@d1#id’ 파라미터의 배열 순서대로 가져와서 Map 과 List 를 이용하여 저장합니다.

## 2.1.3.3. 실제 네트워크 결과

Name	×	Headers	Preview	Response	Cookies	Timing
sample1...		▶ General				
cleopatr...		▼ Response Headers	view source			
defaults.js		Connection: close				
languag...		Content-Language: en				
user-mo...		Content-Length: 2294				
udc.js		Content-Type: text/html;charset=utf-8				
echarts...		Date: Thu, 08 Feb 2018 07:46:06 GMT				
d3.min.js		Server: Apache-Coyote/1.1				
sample1...		▼ Request Headers	view source			
cleopatr...		Accept: /*/*				
cleopatr...		Accept-Encoding: gzip, deflate, br				
checkboxo...		Accept-Language: ko,en-US;q=0.9,en;q=0.8,ja;q=0.7				
data.do		Connection: keep-alive				
		Content-Length: 4054				
		<u>Content-Type: multipart/form-data; boundary=----WebKitFormBoundary160aeLYeMI3Y8Lew</u>				
		Cookie: _ga=GA1.1.2133655032.1507789993				
		Host: localhost:8080				
		Origin: http://localhost:8080				
		Referer: http://localhost:8080/web-servlet/ui/sample/submission/sample1.clx				
		User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/64.0.3282.140 Safari/537.36				
		X-Requested-With: XMLHttpRequest				
		▼ Request Payload				
		-----WebKitFormBoundary160aeLYeMI3Y8Lew				

## 2.1.3.4. 실제 네트워크의 request data

```

-----WebKitFormBoundarySkbDpA9BG5a9I9T0
Content-Disposition: form-data; name="param0"

12
-----WebKitFormBoundarySkbDpA9BG5a9I9T0
Content-Disposition: form-data; name="param1"

32
-----WebKitFormBoundarySkbDpA9BG5a9I9T0
Content-Disposition: form-data; name="@d1#condition"

true
-----WebKitFormBoundarySkbDpA9BG5a9I9T0
Content-Disposition: form-data; name="@d#"

@d1#
-----WebKitFormBoundarySkbDpA9BG5a9I9T0
Content-Disposition: form-data; name="@d1#"

dm_request
-----WebKitFormBoundarySkbDpA9BG5a9I9T0
Content-Disposition: form-data; name="@d2#id"

id1
-----WebKitFormBoundarySkbDpA9BG5a9I9T0
Content-Disposition: form-data; name="@d2#first_name"

first_name1
-----WebKitFormBoundarySkbDpA9BG5a9I9T0
Content-Disposition: form-data; name="@d2#last_name"

last_name1
-----WebKitFormBoundarySkbDpA9BG5a9I9T0
Content-Disposition: form-data; name="@d2#email"

email1
-----WebKitFormBoundarySkbDpA9BG5a9I9T0
Content-Disposition: form-data; name="@d2#gender"

gender1
-----WebKitFormBoundarySkbDpA9BG5a9I9T0
Content-Disposition: form-data; name="@d2#ip_address"

ip_address1
-----WebKitFormBoundarySkbDpA9BG5a9I9T0
Content-Disposition: form-data; name="@d2#sts"

null
-----WebKitFormBoundarySkbDpA9BG5a9I9T0
Content-Disposition: form-data; name="@d2#id"

id2
-----WebKitFormBoundarySkbDpA9BG5a9I9T0
Content-Disposition: form-data; name="@d2#first_name"

first_name2
-----WebKitFormBoundarySkbDpA9BG5a9I9T0
Content-Disposition: form-data; name="@d2#last_name"

last_name2
.....(생략)

```

## 2.2. response

서버로부터 전송 받는 데이터 방식은 text, javascript, blob, filedownload 를 지원합니다.

타입	설명
text	서버로부터 데이터를 받아 데이터셋, 데이터맵에 저장합니다.
javascript	서버로부터 받은 javascript 를 실행합니다.
blob	서버로부터 받은 스트림 데이터를 파일로 다운받습니다.
filedownload	서버로부터 파일을 다운받습니다.

### 2.2.1. text

텍스트로 데이터를 받습니다. 서버미션에서 확인 할 수 있는 데이터 양식으로 수신되면 데이터셋과 데이터맵에 데이터가 저장됩니다. 콘텐츠 타입은 2 가지를 지원합니다.

	내용
<b>Response Header Content-Type</b>	application/json 또는 text/tab-separated-values
<b>수신 후 처리</b>	수신 받은 데이터를 데이터셋, 데이터맵에 저장합니다.
<b>수신 데이터</b>	{       "ds1": [         {           "id": 1,           "first_name": "Cedric",           "last_name": "MacHostie",           "email": "cmachos tie0@amazonaws.com",           "gender": "Male",           "ip_address": "168.86.118.226",           "comp ay_name": "Topicshots",           "dept": "Marketing",           "title": "Ms",           "lang": "Romanian"         },         {           "id": 2,           "first_name": "Virgie",           "last_name": "Goneau",           "email": "vgoneau1@harvard.edu",           "gender": "Male",           "ip_address": "247.173.148.121",           "compay_name": "Rhybox",           "dept": "Accounting",           "title": "Rev",           "lang": "Aymara"         }       ]     }

#### 2.2.1.1. application/json

## 데이터셋

{“UI 데이터셋의 ID”: [{“데이터셋의 컬럼명”: 컬럼의 값”, …… }, {“컬럼명”: 컬럼의 값”, …… }, …… ] }

## 데이터 맵

$$\{\text{"UI 데이터 맵의 ID": \{\text{"데이터 맵의 컬럼명": "컬럼의 값", \dots\}\}}$$

## 데이터셋+데이터맵

```
{“UI 데이터셋의 ID”: [{“데이터셋의 컬럼명”: “컬럼의 값”, …… },
                        {“컬럼명”: “컬럼의 값”, …… }, …… ],
“UI 데이터셋의 ID”: [{“컬럼명”: “컬럼의 값”, …… },
                        {“컬럼명”: “컬럼의 값”, …… }, …… ],
“UI 데이터맵의 ID”: {“컬럼명”: “컬럼의 값”, …… },
“UI 데이터맵의 ID”: {“컬럼명”: “컬럼의 값”, …… } }
```

## 2.2.1.2. Response Header Content-Type: application/json 전송 결과

The screenshot shows the 'Headers' tab in a web browser's developer tools. The 'Response Headers' section is expanded, showing the following headers:

- Content-Type:** application/json (highlighted with a red box)
- Date:** Thu, 08 Feb 2018 08:07:44 GMT
- Server:** Apache-Coyote/1.1
- Transfer-Encoding:** chunked

The 'Request Headers' section is also visible below, showing headers like Accept, Accept-Encoding, Accept-Language, Connection, Content-Length, Content-Type, Cookie, Host, Origin, and Referer.

## 2.2.1.3. Response Header Content-Type: application/json 전송 데이터

The screenshot shows the 'Response' tab in a web browser's developer tools. The response body is displayed as a JSON object, with the first part of the array highlighted in red:

```
{
  "ds_sample": [
    {
      "id": 1,
      "first_name": "Cedric",
      "last_name": "MacHostie",
      "email": "cmachostie0@amazon"
    },
    {
      "id": 2,
      "first_name": "Virgie",
      "last_name": "Goneau",
      "email": "vgoneau1@harvard.edu"
    }
  ]
}
```

## 2.2.1.4. 전송 데이터 상세

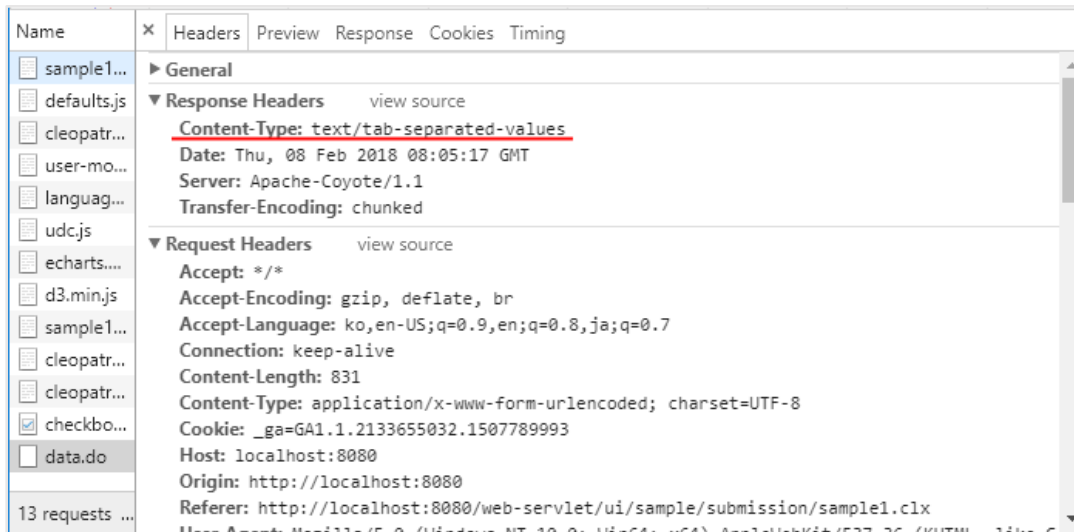
```
{
  "ds_sample": [
    {
      "id": 1,
      "first_name": "Cedric",
      "last_name": "MacHostie",
      "email": "cmachostie0@amazon",
      "gender": "Male",
      "ip_address": "168.86.118.226",
      "compay_name": "Topicshots",
      "dept": "Marketing",
      "title": "Ms",
      "lang": "Romanian"
    },
    {
      "id": 2,
      "first_name": "Virgie",
      "last_name": "Goneau",
      "email": "vgoneau1@harvard.edu",
      "gender": "Male",
      "ip_address": "247.173.148.121",
      "compay_name": "Rhybox",
      "dept": "Accounting",
      "title": "Rev",
      "lang": "Aymara"
    }
  ]
}
```

### 2.2.1.5. text/tab-separated-values

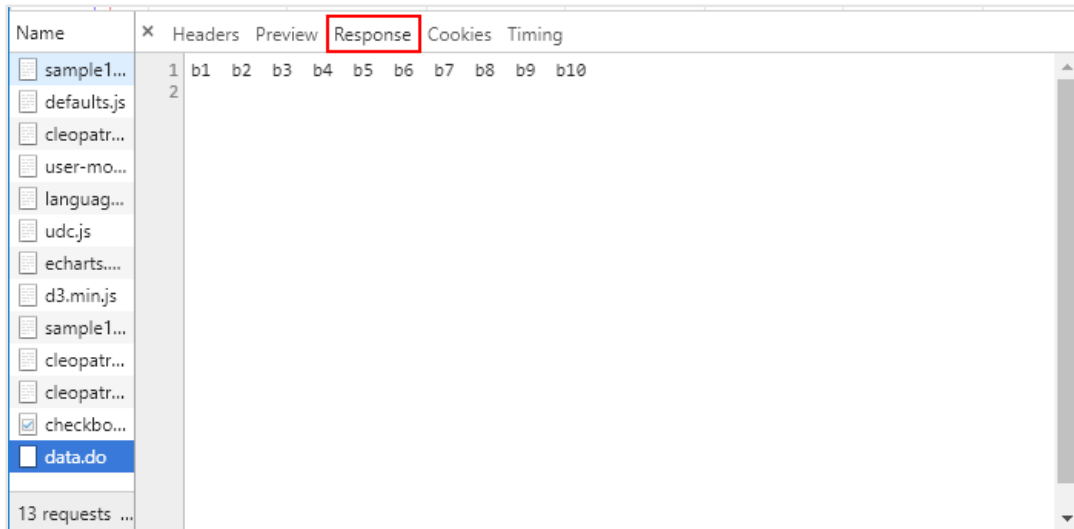
대용량의 그리드 데이터를 받을 때 사용하는 방식입니다. 하나의 데이터셋에만 사용이 가능합니다.

```
value[tab]value[tab]value
value[tab]value[tab]value
.....
```

### 2.2.1.6. Response Header Content-Type: text/tab-separated-values 전송 결과



### 2.2.1.7. Response Header Content-Type: text/tab-separated-values 전송 데이터

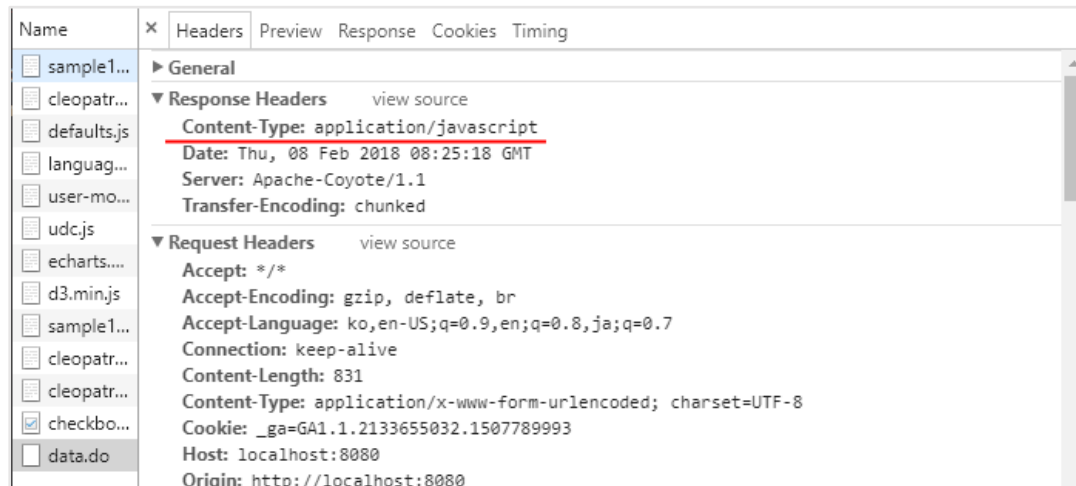


## 2.2.2. javascript

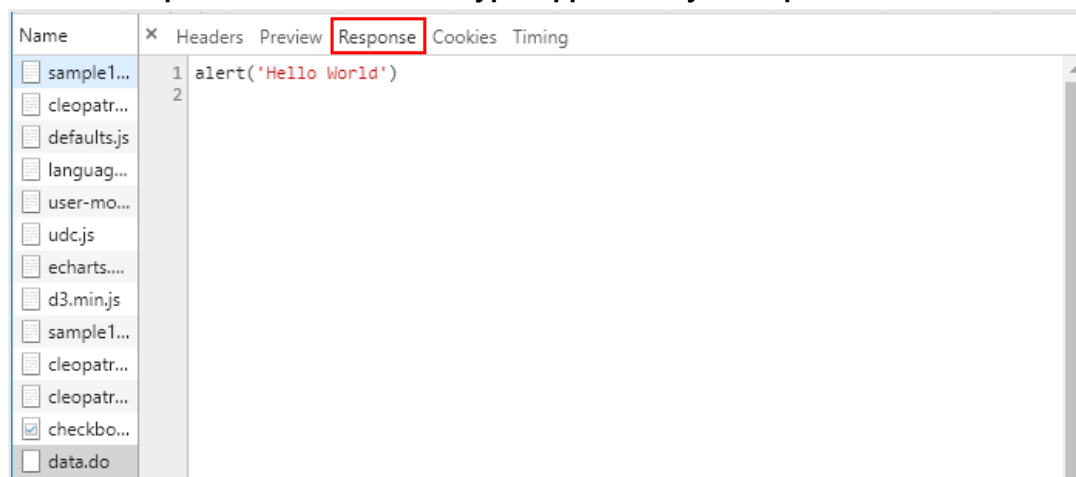
서버에서 전송된 자바스크립트의 텍스트를 실행합니다.

	내용
<b>Response Header Content-Type</b>	application/javascript
<b>수신 후 처리</b>	수신 받은 자바스크립트의 텍스트를 eval 합니다.
<b>수신 데이터</b>	<pre>“function message(msg){     alert(msg); } alert(message(‘Hello World’));”</pre>

### 2.2.2.1. Response Header Content-Type: application/javascript 전송 결과



### 2.2.2.2. Response Header Content-Type: application/javascript 전송 데이터



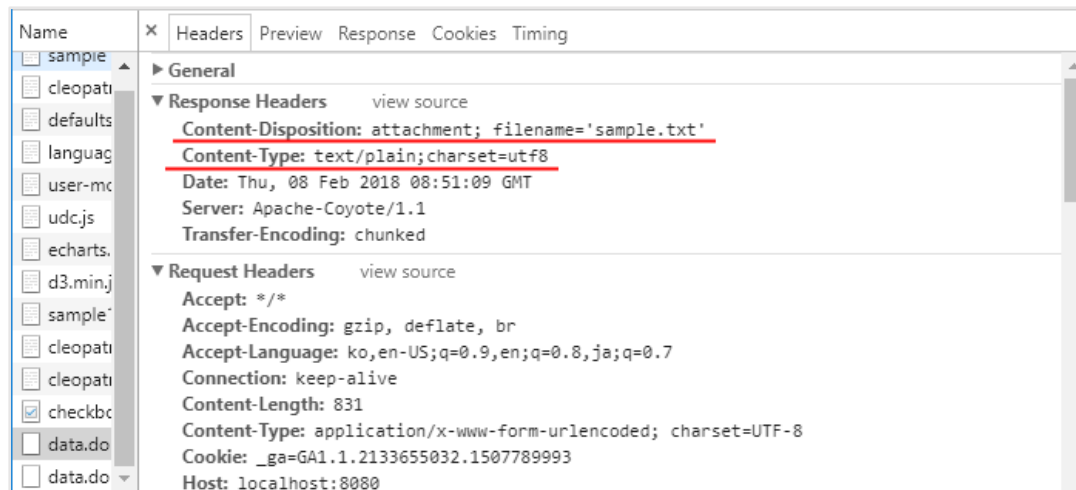


### 2.2.3. blob

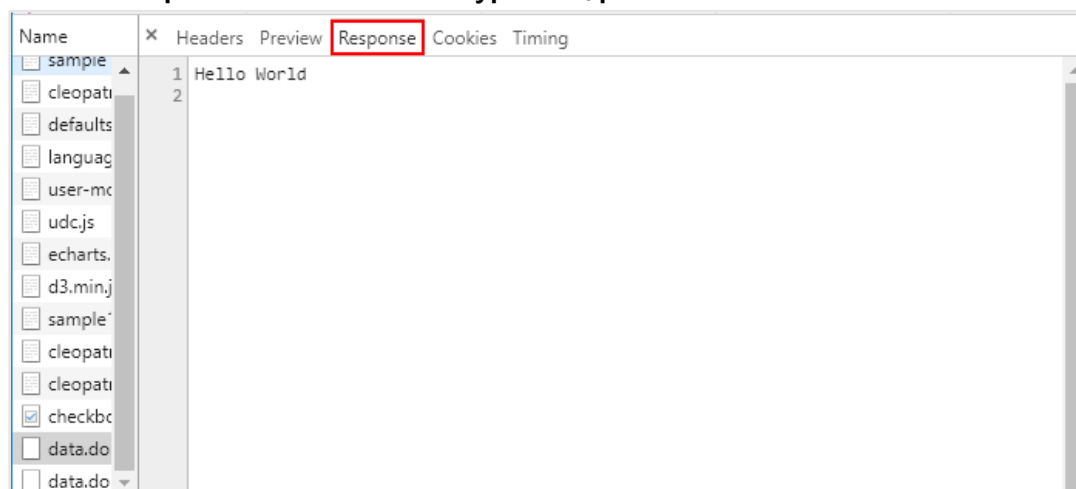
서버에서 blob 으로 된 데이터를 받습니다. 작은 크기의 파일(이미지, 텍스트)을 다운로드 할 때 사용합니다.

	내용
<b>Response Header Content-Type</b>	파일에 대한 mime type
<b>Response Header Content-Disposition</b>	attachment; filename='파일이름.확장자'
<b>수신 후 처리</b>	blob 된 데이터를 파일로 가공하여 다운로드를 발생시킵니다.
<b>수신 데이터</b>	blob 으로된 데이터

#### 2.2.3.1. Response Header Content-Type: text/plain 전송 결과



#### 2.2.3.2. Response Header Content-Type: text/plain 전송 데이터



#### 2.2.4. filedownload

클라이언트에서 처리가 없으며 서버에서 바로 파일을 받을 때 사용합니다. 대용량 파일에 대한 다운로드에 사용됩니다.

	내용
Response Header Content-Type	없음.
수신 후 처리	없음.
수신 데이터	없음.