

## Chapter2. 데이터 전송

Hello! world

# JSP/Servlet

임명진 연구원



## 1. 서버프로그램은 왜 필요한가?

**정적 웹서버는 클라이언트의 요청에 따른  
모든 경우의 수를 고려한 HTML문서들이 저장되어 있어야 한다.**



## 1. 서버프로그램은 왜 필요한가?

정적 웹서버는 클라이언트의 요청에 따른  
모든 경우의 수를 고려한 HTML 문서들이 저장되어 있어야 한다.



개발자들의 시간과 노동력 🤖 🌀 😡



## 1. 서버프로그램은 왜 필요한가?

정적 웹서버는 클라이언트의 요청에 따른  
모든 경우의 수를 고려한 HTML문서들이 저장되어 있어야 한다.

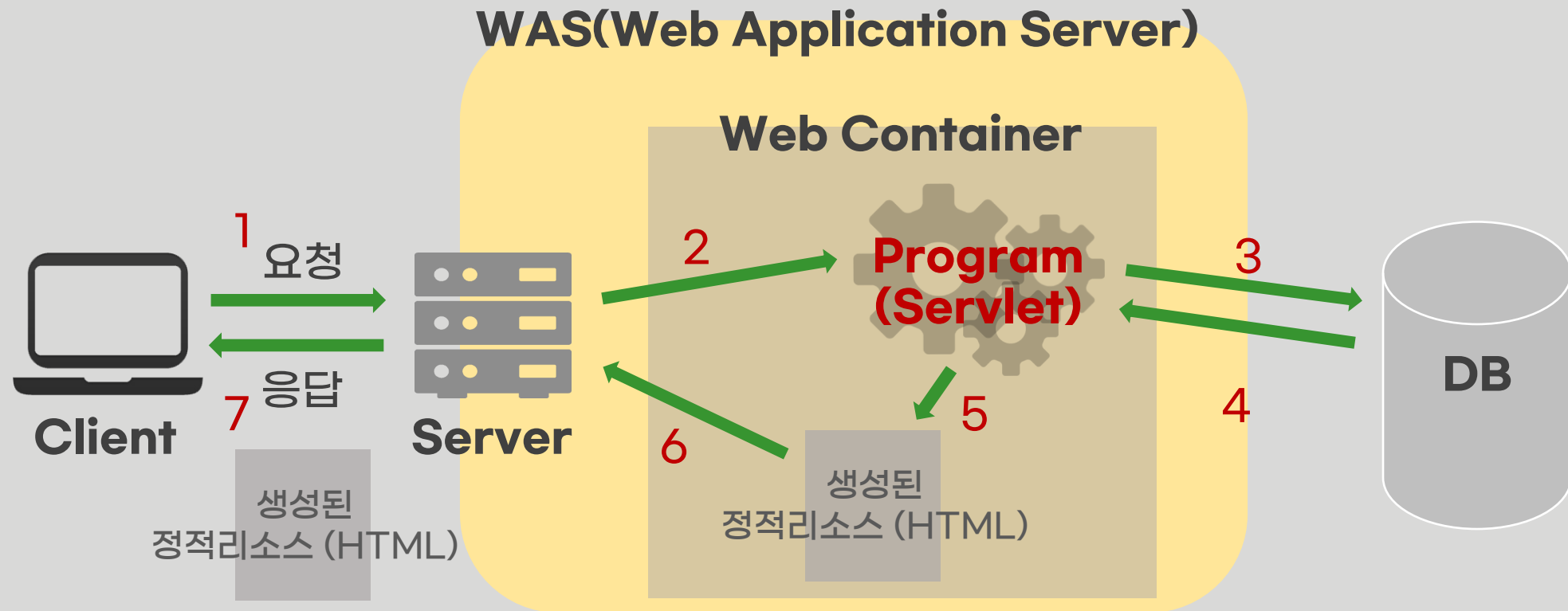


개발자들의 시간과 노동력 🤖 🌀 😡

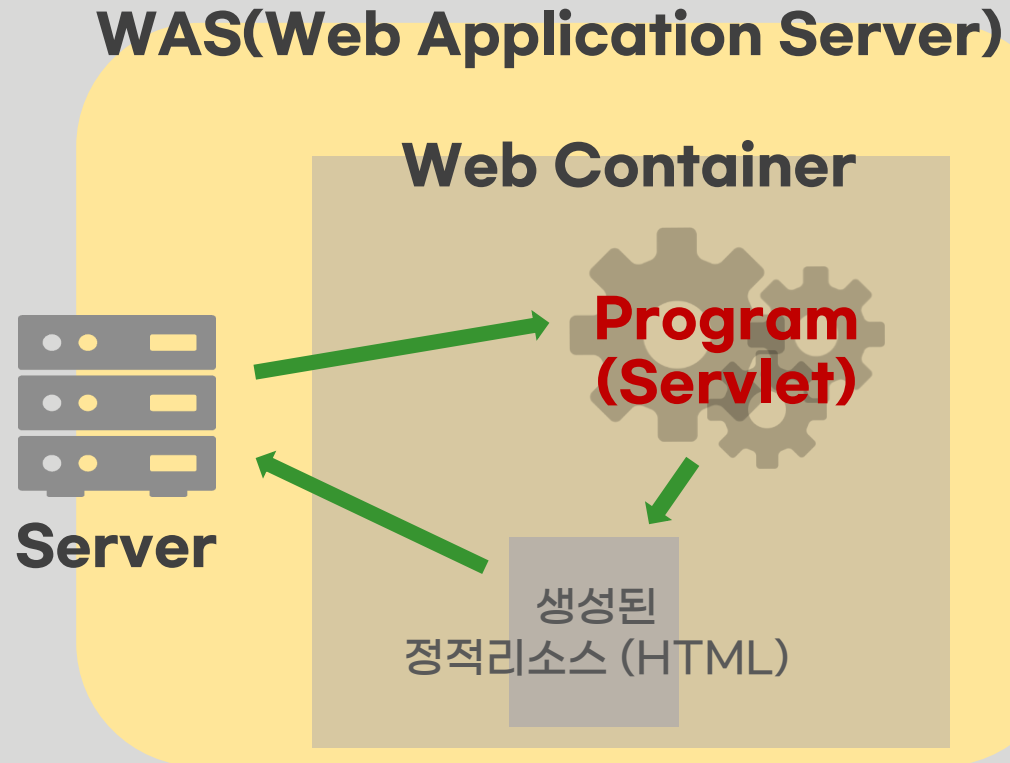


클라이언트 요청정보를 담고 있는 HTML문서를  
만들어 주는 서버 프로그램이 필요하다!

## 2. Servlet의 구조



## 2. Servlet의 구조



### Web Server

- : 정적인 콘텐츠를 제공하는 서버
- : 요청을 컨테이너로 전달하고 결과를 넘겨주는 역할

### WAS(웹 서버+웹컨테이너)

- : 동적으로 콘텐츠를 생성하여 제공하는 서버



### Web Container

- : JSP와 Servlet을 실행시킬 수 있는 SW



## 3. Servlet의 특징

- **.java** 확장자를 가짐
- **Java Multi Thread**를 이용하여 동작함 -> 속도와 메모리 면에서 효율적임
- 객체지향적 -> 대규모 Application 개발에 적합함
- **HttpServlet** 클래스를 상속받음



## 4.URL Mapping

### URL Mapping :

Web browse에서 Servlet을 동작시키기 위해 실제 Java 클래스의 이름 대신, Servlet을 요청하기 위한 문자열을 Servlet 클래스와 Mapping(매핑) 시키는 것

원래주소 : `http://localhost:8081/FirstProject/Servlet/HelloWeb`

매핑된 주소 : `http://localhost:8081/FirstProject/hweb`

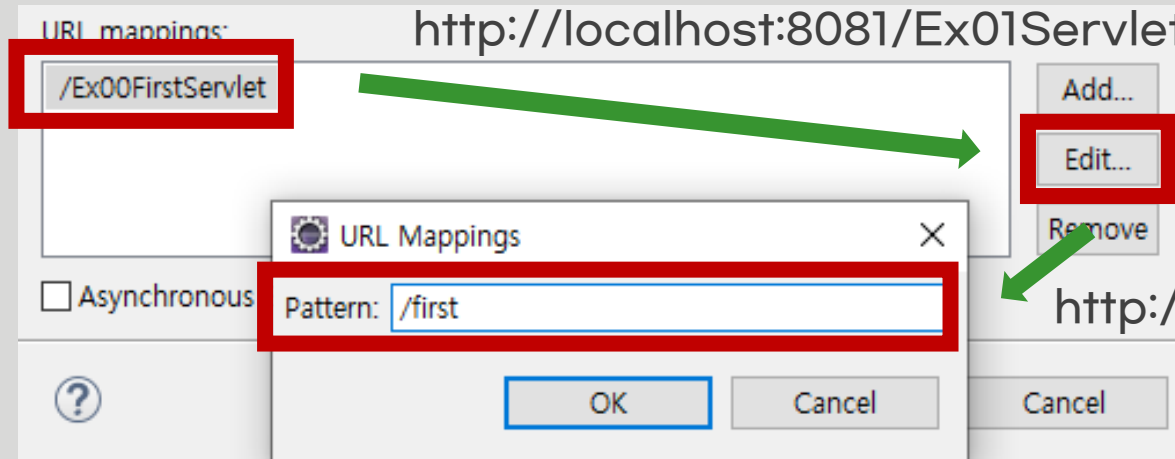
주소가 길다 -> 불편함  
경로가 드러난다 -> 보안상 문제



## 4.URL Mapping

### URL Mapping 방법 1. **annotation** 사용

기본값 : class 명과 동일



http://localhost:8081/Ex01Servlet/**first**

## 4.URL Mapping

### URL Mapping 방법 2. annotation 사용

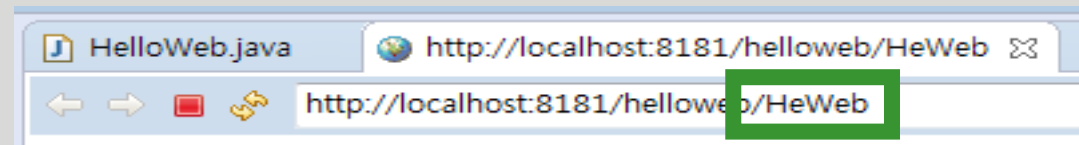
```
package com.javalec.ex;

import java.io.IOException;

/**
 * Servlet implementation class HelloWeb
 * @WebServlet("/HeWeb")
 */
public class HelloWeb extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public HelloWeb() {
        super();
        // TODO Auto-generated constructor stub
    }
}
```

@WebServlet("/매핑한문자열")





## 4.URL Mapping

### URL Mapping 방법 3. **web.xml** 사용

```
<servlet>
  <servlet-name>helloworld</servlet-name>
  <servlet-class>com.servlet.helloworld</servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>helloworld</servlet-name>
  <url-pattern>/hello</url-pattern>
</servlet-mapping>
```

**servlet-name** : servlet 별명

**servlet-class** : servlet class의 실제 경로  
(package 경로)

**url-pattern** : URL 매핑 값 작성



## 5. 서블릿 생명주기

### 서블릿 생명주기(Life Cycle)

: 서블릿 객체 생성부터 소멸까지의 과정

**init()**

초기화



**service()**

요청처리



**destroy()**

소멸

## 5. 서블릿 생명주기

### 상속 메서드 선택

1) 클라이언트로 부터 **최초** 요청이 들어오면 Servlet 클래스가 로딩되어  
요청에 대한 Servlet 객체 생성

생성자 호출 -> Servlet 으로서의 역할을 하진 못함

init()

2) Servlet 초기화 (Servlet 으로서의 역할 수행)

비용이 많은 작업 이므로 서블릿이 처음 생성되었을 때 딱 한번만 호출 (자원절약)

service()

doGet()

3) HTTP 요청 메서드에 따라서 doGet() or doPost() 호출

doPost()

서블릿 객체가 생성되어 있을 경우에는 service만 호출

destroy()

4) Servlet 객체 소멸 (ex. 서버 종료, 재시작 등)

서버 종료 시 딱 한번만 호출



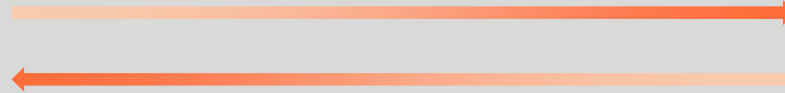
## Contents

1. Form 태그를 사용하여 데이터를 전송하는 예제를 작성할 수 있다.
2. Request 와 Response 객체를 사용하여 요청/응답 처리를 할 수 있다.
3. 응답 데이터의 인코딩 방식을 이해한다.

## 데이터 전송



요청(request)  
(사용자가 검색한 데이터)



응답(response)  
(사용자가 검색한 결과)



Server

## 데이터 전송

```
<form id="sform" name="sform" action="https://search.naver.com/search.naver" method="get" role="search">
  <fieldset>
    <legend class="blind">검색</legend>
    <select id="where" name="where" title="검색 범위 선택" class="blind">...</select>
    <input type="hidden" id="sm" name="sm" value="top_hyt">
    <input type="hidden" id="fbm" name="fbm" value="0">
    <input type="hidden" id="acr" name="acr" value disabled="">
    <input type="hidden" id="acq" name="acq" value disabled="">
    <input type="hidden" id="qdt" name="qdt" value disabled="">
    <input type="hidden" id="ie" name="ie" value="utf8">
    <input type="hidden" id="acir" name="acir" value disabled="">
    <input type="hidden" id="os" name="os" value disabled="">
    <input type="hidden" id="bid" name="bid" value disabled="">
    <input type="hidden" id="pkid" name="pkid" value disabled="disabled">
    <input type="hidden" id="eid" name="eid" value disabled="disabled">
    <input type="hidden" id="mra" name="mra" value disabled="disabled">
    <div class="green_window" style>...</div>
    <button id="search_btn" type="submit" title="검색" tabindex="3" class="btn_submit" onclick="window.nclick(this,'sch.action','','',event);" style>
      <span class="blind">검색</span>
      <span class="ico_search_submit"></span>
    </button>
  </fieldset>
</form>
```

\* <form> 태그를 사용한 데이터 전송 필수 조건  
action, method, name, submit(button)

실제 html 코드





## 데이터 전송

확인하기

<form> 태그 활용 데이터를 전송할 때 필요한 조건은?

요청 방식 (get/post)  
(생략 시 기본값은 get)

요청 경로

```
<form action="Ex01DataSend" method="get">  
  <input type="text" name="id">  
  <input type="password" name="pw">  
  <input type="submit" value="Login">  
</form>
```

식별자

제출 버튼 : 클릭하는 순간 서버로 데이터 전송



## 데이터 전송

확인하기

<form> 태그 활용 데이터를 전송할 때 필요한 조건은?

```
<form [ ] [ ] >
  <input type="text" [ ] >
  <input type="password" [ ] >
  <input [ ] value="login">
</form>
```

## 데이터 전송 실습1

Ex1-1

1개의 데이터를 입력하여 서버로 전송할 수 있는 <form> 태그를 완성하시오.

DATA :

SEND

Source Ex01DataSend.html

\* 요청 경로는 브라우저의 URL 주소창을 보면 가장 좋음!

절대 경로 : http://localhost:8081/Ex02DataSend/Ex01DataSend 로 작성

상대 경로 : Ex01DataSend 로 작성

```
<form action="Ex01DataSend">
  DATA : <input type="text" name="data">
  <input type="submit" value="SEND">
</form>
```

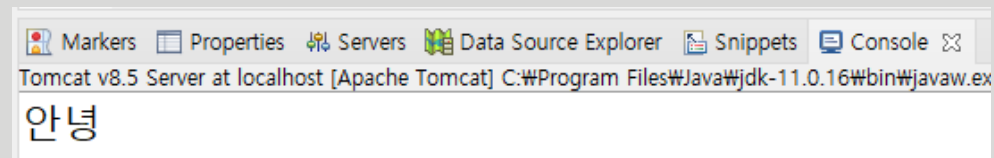
ore > tmp0 > wtpwebapps > Ex02DataSend	
이름	수정한
META-INF	2022-1
WEB-INF	2022-1
Ex01DataSend	2022-1



## 데이터 전송 실습1

Ex1-2

요청 데이터를 받아 콘솔창에 출력하시오.



\* html 에서 데이터 작성 후 Ex01DataSend로 제출 시 콘솔창에 데이터 출력 확인

Source Ex01DataSend.java

```
String data = request.getParameter("data");
```

“data” 로 식별할 수 있는 데이터 가져와서 변수에 저장

```
System.out.println(data); 콘솔창에 출력
```



## request.getParameter()

```
String data = request.getParameter("data");
```

getParameter()의  
반환타입

요청객체

지정한 파라미터 값 반환

식별자

getParameter(식별자(String))

<input> 태그에 작성된  
각 name값 문자열로 작성

: request(요청) 객체가 가지고 있는 요청 파라미터 값 반환(String) 메서드

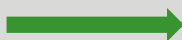


## 데이터 전송 실습1

Ex1-3

요청 데이터를 받아 화면(브라우저)에 출력하시오.

Ex01DataSend.html



출력해보자

Ex01DataSend.java



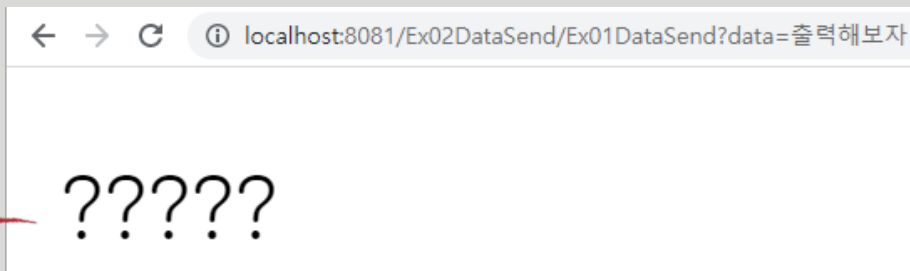
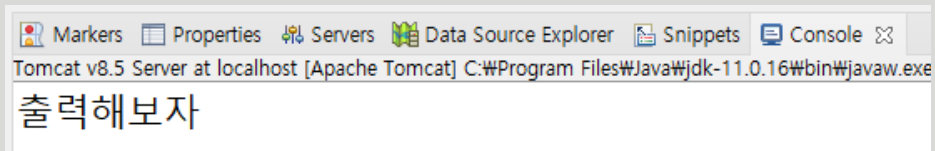


## 데이터 전송 실습1

Ex1-3

요청 데이터를 받아 화면(브라우저)에 출력하시오.

Ex01DataSend.html



Ex01DataSend.java



콘솔창에서는 잘 출력되지만 화면상에는 잘 출력되지 않음 (응답데이터 인코딩이 필요!)



## 응답 데이터 인코딩하기

### `response.setCharacterEncoding("UTF-8")`

응답 데이터 인코딩 (생략가능)

But, 브라우저마다 인코딩 방식이 다르기 때문에 한글 데이터를 출력하기 위해서는  
응답 데이터의 인코딩 방식을 브라우저에게 알려줘야 함

### `response.setContentType("text/html; charset=UTF-8")`

브라우저에게 응답하는 데이터의 형식을 알려주기 위함

text/html : 응답하는 데이터가 텍스트 또는 html 형식임

charset=UTF-8 : 인코딩 방식이 UTF-8임





## 데이터 전송 실습1

Ex1-3

요청 데이터를 브라우저 상에 출력하시오.

Source Ex01DataSend.java

```
response.setCharacterEncoding("UTF-8");
```

```
response.setContentType("text/html; charset=UTF-8");
```

브라우저에게 응답 데이터 형식 및 인코딩 방식 알려줌

```
PrintWriter out = response.getWriter();
```

```
out.print(data);
```

 텍스트 출력 스트림을 통해 데이터 출력

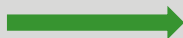


## 데이터 전송 실습2

Ex2

서버로 전송한 숫자 1개를 화면에 다음과 같이 출력하시오.

Ex02Number.html



localhost:8081/Ex02DataSend/Ex02Number?num=1212

입력한 숫자는 **1212**입니다

Ex02Number.java



## 데이터 전송 실습3

Ex3

서버로 전송한 숫자 2개를 활용해 화면에 다음과 같은 연산식으로 출력하시오.

+

SEND

Ex03Plus.html



localhost:8081/Ex02DataSend/Ex03Plus?num1=3&amp;num2=5

 $3 + 5 = 8$ 

Ex03Plus.java





## 데이터 전송 실습3

Ex3

서버로 전송한 숫자 2개를 활용해 화면에 다음과 같은 연산식으로 출력하시오.

Source Ex03Plus.java

```
String num1 = request.getParameter("num1");
```

String 형태로는 숫자 연산을 할 수 없음

```
int int_num1 = Integer.parseInt(num1);
```

```
int int_num2 = Integer.parseInt(request.getParameter("num2"));
```

Integer 클래스의 parseInt() 메서드 활용 String 을 int 형으로 변환할 수 있음

**\* 이때 메서드의 인자(매개변수)는 숫자로만 이루어진 문자열 이어야 함!**



## 데이터 전송 실습4

Ex4

서버로 전송한 숫자 2개와 연산기호를 활용해 화면에 다음과 같은 연산식으로 출력하시오.

15	* ▾	3	SEND
----	-----	---	------

Ex04Operation.html



← → ↻ ⓘ localhost:8081/Ex02DataSend/Ex04Operation?num1=15&ope=\* &num2=3

$$15 * 3 = 45$$

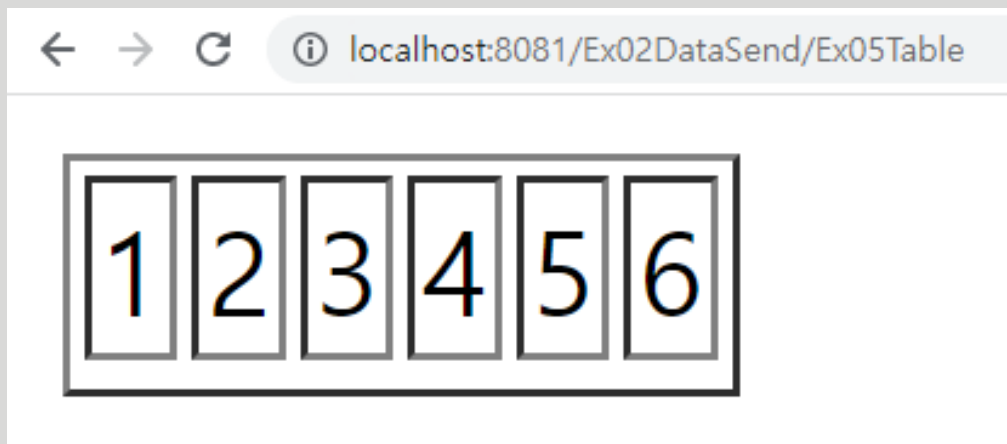
Ex04Operation.java



## 데이터 전송 실습5

Ex5

화면에 다음과 같은 테이블을 출력하시오.



The screenshot shows a web browser window with the address bar displaying 'localhost:8081/Ex02DataSend/Ex05Table'. Below the address bar, there is a table with 6 columns and 1 row, containing the numbers 1 through 6.

1	2	3	4	5	6
---	---	---	---	---	---

Ex05Table.java



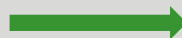
## 데이터 전송 실습6

**Ex6**

서버로 숫자 1개를 전송해 해당 숫자만큼 셀(cell)을 가진 테이블을 출력하시오.

Ex06MakeTable.html



← → ↻ ⓘ localhost:8081/Ex02DataSend/Ex06MakeTable?cell=8

1	2	3	4	5	6	7	8
---	---	---	---	---	---	---	---

Ex06MakeTable.java





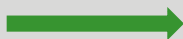
## 데이터 전송 실습7

Ex7

색상을 선택한 후 서버로 전송하면 해당 색상을 배경색으로 하는 화면을 출력하시오.



Ex07Bgcolor.html



Ex07Bgcolor.java





## 데이터 전송 실습8

Ex8

색상 1개와 2개의 숫자를 서버로 전송하면 해당 색상을 배경색으로 하면서  
입력한 숫자 만큼의 범위를 가지는 구구단 표를 출력하시오. (ex. 3,6 입력 -> 3단~6단)



3

에서

6

까지

SEND

Ex08Gugu.html



localhost:8081/Ex02DataSend/Ex08Gugu?color=%23b778ce&num1=3&num2=6

3 * 1 = 3	3 * 2 = 6	3 * 3 = 9	3 * 4 = 12	3 * 5 = 15	3 * 6 = 18	3 * 7 = 21	3 * 8 = 24	3 * 9 = 27
4 * 1 = 4	4 * 2 = 8	4 * 3 = 12	4 * 4 = 16	4 * 5 = 20	4 * 6 = 24	4 * 7 = 28	4 * 8 = 32	4 * 9 = 36
5 * 1 = 5	5 * 2 = 10	5 * 3 = 15	5 * 4 = 20	5 * 5 = 25	5 * 6 = 30	5 * 7 = 35	5 * 8 = 40	5 * 9 = 45
6 * 1 = 6	6 * 2 = 12	6 * 3 = 18	6 * 4 = 24	6 * 5 = 30	6 * 6 = 36	6 * 7 = 42	6 * 8 = 48	6 * 9 = 54

Ex8Gugu.java

## Chapter2. 데이터 전송

*next*

# ch3. Get/Post

강예진 연구원