2️⃣

Section2. 서블릿

프로젝트 생성

프로젝트 정보

Name: servlet

Java: 17

Packaging: War(JSP의 경우 War를 선택해야 한다)

Dependencies

- `Spring web`
- `Lombok`

Hello 서블릿

스프링 부트는 톰캣 서버를 내장하고 있으므로, 톰캣 서버 설치 없이 편리하게 서블릿 코드를 실행할 수 있다.

> 💡 실습 코드 작성

서블릿 등록

- 서블릿을 직접 등록해서 사용할 수 있게 하는 어노테이션

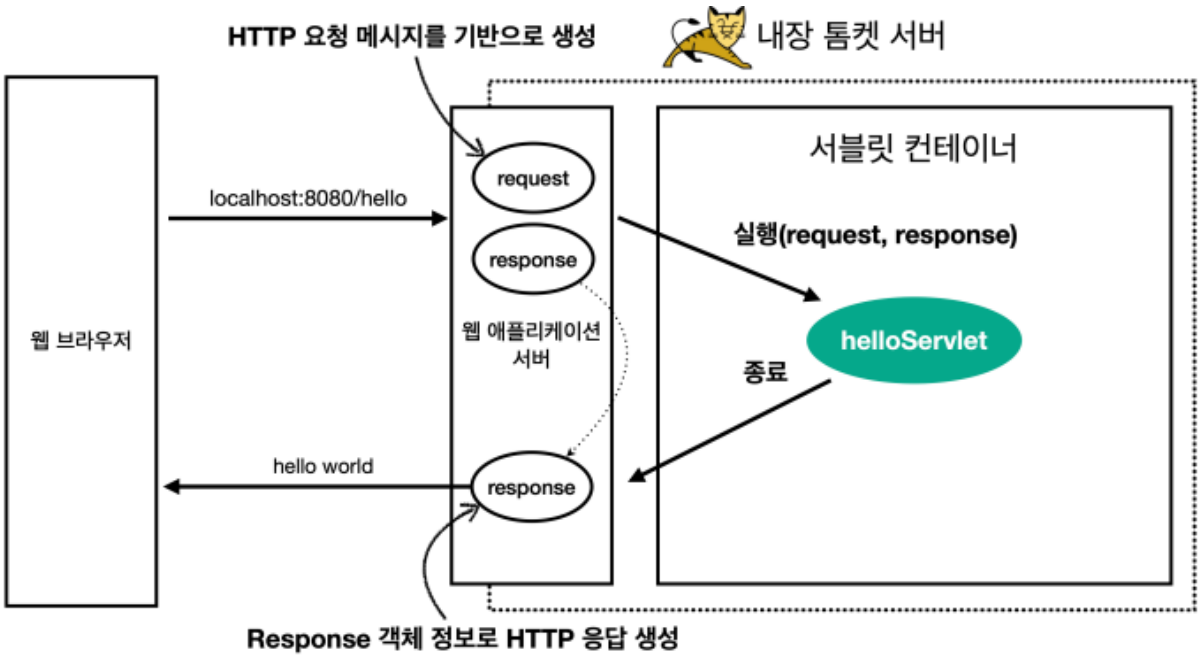
```
@ServletComponentScan
@SpringBootApplication
public class ServletApplication {
    public static void main(String[] args) {
        SpringApplication.run(ServletApplication.class, args)
    }
}
```

```
// 서블릿 애노테이션, name: 서블릿 이름, urlPatterns: URL 매핑
@WebServlet(name = "helloServlet", urlPatterns = "/hello")
public class HelloServlet extends HttpServlet {
    @Override
    protected void service(HttpServletRequest request, HttpSe
        System.out.println("HelloServlet.service");
        System.out.println("request = " + request);
        System.out.println("response = " + response);

        String username = request.getParameter("username");
        System.out.println("username = " + username);

        response.setContentType("text/plain");
        response.setCharacterEncoding("utf-8");
        response.getWriter().write("hello " + username);
    }
}
```

HTTP 요청 메시지 로그로 확인하기

```
// application.properties
logging.level.org.apache.coyote.http11=debug
```

서블릿 컨테이너 동작 방식



💡 강의에서 나오는 html 파일의 경우 복사-붙여넣기 진행

HttpServletRequest - 개요

HttpServletRequest 역할

서블릿은 개발자가 HTTP 요청 메시지를 편리하게 사용할 수 있도록 개발자 대신 HTTP 요청 메시지 파싱, 그 결과를 `HttpServletRequest` 객체에 담아서 제공한다.

```
<!-- HTTP 요청 메시지 -->
POST /save HTTP/1.1
Host: localhost:8080
Content-Type: application/x-www-form-urlencoded
username=kim&age=20
```

START LINE

- HTTP 메서드
- URL
- 쿼리 스트링
- 스키마, 프로토콜

헤더

- 헤더 조회

바디

- form 파라미터 형식 조회
- message body 데이터 직접 조회

부가기능

임시 저장소 기능

- 해당 HTTP 요청이 시작부터 끝날 때 가지 유지되는 임시 저장소 기능
  - 저장: `request.setAttribute(name, vlaue)`
  - 조회: `requset.getAttribute(name)`

세션 관리 기능

- request.getSession(create: true);

> 💡 HttpServlet— 를 사용할 때 중요한 것은 이 객체들이 HTTP 요청 메시지, HTTP 응답메시지를 편리하게 사용하도록 도와주는 객체라는 점이다.
>
> 이 기능에 대해서 깊이있는 이해를 하려면 “
> **HTTP 스펙이 제공하는 요청, 응답 메시지 자체를 이해**” 해야 한다.

HttpServletRequest - 기본 사용법

> 💡 해당 강의에서는 `HttpServletRequest` 객체에서 얻을 수 있는 정보를 출력하는 실습 진행

HTTP 요청 데이터 - 개요

HTTP 요청 메시지를 통해 클라이언트에서 서버로 데이터를 전달하는 방법

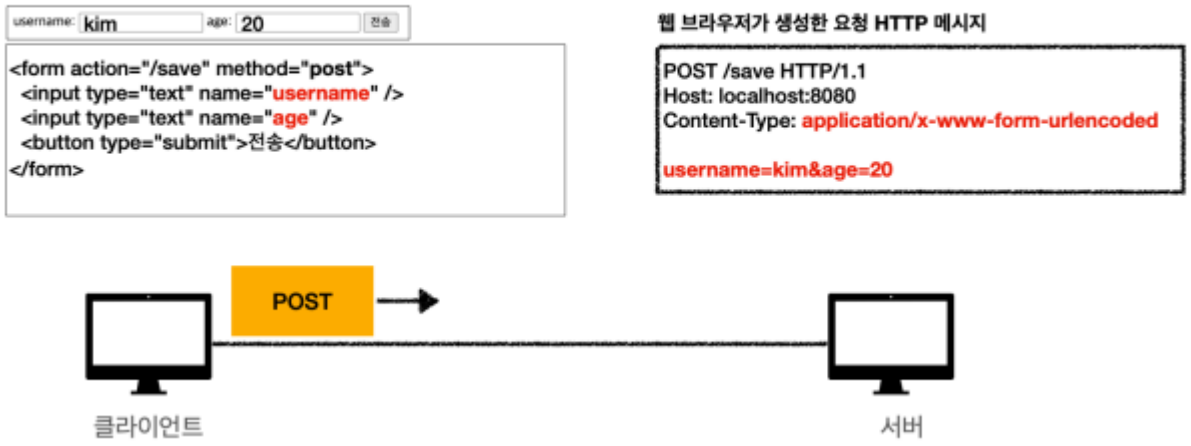
GET - 쿼리 파라미터

- /url?**username=hello&age=20**

- 메시지 바디 없이, URL의 쿼리 파라미터에 데이터를 포함해서 전달

  예) 검색, 필터, 페이징등에서 많이 사용

POST - HTML Form



- `content-type` : `application/x-www-form-urlencoded`
- 메시지 바디에 **쿼리 파라미터** 형식으로 전달 username=hello&age=20

  예) 회원 가입, 상품 주문, HTML Form 사용

HTTP message body

- HTTP API에서 주로 사용, `JSON`, `XML`, `TEXT`
- 데이터 형식은 주로 JSON 사용
- POST, PUT, PATCH

HTTP 요청 데이터 - GET 쿼리 파라미터

전달 데이터

- username = hello
- age = 20

쿼리 파라미터는 URL에 다음과 같이 `?`를 시작으로 보낼 수 있다.

- 추가 파라미터는 `&`로 구분하면 된다.

쿼리 파라미터 조회 메서드

```
// 단일 파라미터 조회
String username = request.getParameter("username");

// 파라미터 이름들 모두 조회
Enumeration<String> parameterNames = request.getParameterName

// 파라미터를 Map으로 조회
Map<String, String[]> parameterMap = request.getParameterMap(

//복수 파라미터 조회
String[] usernames = request.getParameterValues("username");
```

💡 **※ 복수 파라미터에서 단일 파라미터 조회 ※**

`username=hello&username=kim` 과 같이 파라미터 이름은 하나인데, 값이 중복이면?

`request.getParameter()` 는 하나의 파라미터 이름에 대해서 단 하나의 값만 있을 때 사용해야 한다. 지금처럼 중복일 때는 `request.getParameterValues()` 를 사용해야 한다.

참고로 이렇게 중복일 때 `request.getParameter()` 를 사용하면 `request.getParameterValues()` 의 첫번째 값을 반환한다.

HTTP 요청 데이터 - POST HTML Form

특징

- content-type: `'application/x-www-form-urlencoded'`
- 메시지 바디에 쿼리 파라미터 형식으로 데이터를 전달한다.
  - 'username=hello&age=20'

💡 실습 코드 작성

POST의 HTML Form을 전송하면 웹 브라우저는 다음 형식으로 HTTP 메시지를 만든다.

- 요청 URL: http://localhost:8080/request-param
- content-type: `application/x-www-form-urlencoded`
- message body: `username=hello&age=20`

'application/x-www-form-urlencoded' 형식은 GET에서 살펴본 쿼리 파라미터 형식과 같다. 따라서 **'쿼리 파라미터 조회 메서드를 그대로 사용'** 하면 된다.

클라이언트입장에서는 두 방식에 차이가 있지만, 서버 입장에서는 둘의 형식이 동일하므로, `request.getParameter()` 로 편리하게 구분없이 조회할 수 있다.

정리하면 '`request.getParameter()`' 는 GET URL 쿼리 파라미터 형식도 지원하고, POST HTML Form 형식 둘 다 지원한다.

💡 content-type은 HTTP 메시지 바디의 데이터 형식을 지정한다.

GET URL 쿼리 파라미터 형식으로 클라이언트에서 서버로 데이터를 전달할 때는 HTTP 메시지 바디를 사용하지 않기 때문에 content-type이 없다.

POST HTML Form 형식으로 데이터를 전달하면 HTTP 메시지 바디에 해당 데이터를 포함해서 보내기 때문에 바디에 포함된 데이터가 어떤 형식인지 content-type을 꼭 지정해야 한다. 이렇게 폼으로 데이터를 전송하는 형식을 application/x-www-form-urlencoded 라 한다.

HTTP 요청 데이터 - API 메시지 바디 - 단순 텍스트

HTTP message body에 데이터를 직접 담아서 요청

- HTTP API에서 주로 사용, `JSON`, `XML`, `TEXT`
- 데이터 형식은 주로 `JSON` 사용

- POST, PUT, PATCH

먼저 가장 단순한 텍스트의 메시지를 HTTP 메시지 바디에 담아서 전송하고, 읽어보자.

HTTP 메시지 바디의 데이터를 `InputStream`을 사용해서 직접 읽을 수 있다.

> 💡 실습 코드 작성

```
@WebServlet(name = "requestBodyStringServlet", urlPatterns =
public class RequestBodyStringServlet extends HttpServlet {
    @Override
    protected void service(HttpServletRequest request, HttpSe
        ServletInputStream inputStream = request.getInputStre
        String messageBody = StreamUtils.copyToString(inputSt

        System.out.println("messageBody = " + messageBody);

        response.getWriter().write("ok");
    }
}
```

HTTP 요청 데이터 - API 메시지 바디 - JSON

JSON 형식 전송

- POST http://localhost:8080/request-body.json
- content-type: '`application/json`'
- message body: {"username": "hello", "age": 20}
- 결과: 'messageBody = {"username": "hello", "age": 20}'

JSON 형식 파싱 추가

JSON 형식으로 파싱하기 위한 객체 생성

```
import lombok.Getter;
import lombok.Setter;
```

```
@Getter @Setter
public class HelloData {
     private String username;
     private int age;
}
```

```
@WebServlet(name = "requestBodyJsonServlet", urlPatterns = "/
public class RequestBodyJsonServlet extends HttpServlet {
     private ObjectMapper objectMapper = new ObjectMapper();
     @Override
     protected void service(HttpServletRequest request, HttpS
            ServletInputStream inputStream = request.getInput
            String messageBody = StreamUtils.copyToString(inp
            StandardCharsets.UTF_8);

            System.out.println("messageBody = " + messageBody

            HelloData helloData = objectMapper.readValue(mess
            HelloData.class);

            System.out.println("helloData.username = " + hell
            System.out.println("helloData.age = " + helloData
            response.getWriter().write("ok");
    }
}
```

💡 JSON 결과를 파싱해서 사용할 수 있는 자바 객체로 변환하려면 **Jackson**, **Gson** 같은 JSON 변환 라이브러리를 추가해서 사용해야 한다.

스프링 부트로 Spring MVC를 선택하면 기본으로 **Jackson** 라이브러리 (`ObjectMapper`)를 함께 제공한다.

HttpServletResponse - 기본 사용법

HttpServletResponse 역할

- **HTTP 응답 메시지 생성**
  - HTTP 응답코드 지정
  - 헤더 생성
  - 바디 생성
- **편의 기능 제공**
  - Content-Type, 쿠키, Redirect

HttpServletResponse - 기본 사용법

```
@WebServlet(name = "responseHeaderServlet", urlPatterns = "/r
public class ResponseHeaderServlet extends HttpServlet {
    @Override
    protected void service(HttpServletRequest request, HttpSe
        // [status-line]
        response.setStatus(HttpServletResponse.SC_OK);

        // [response-headers]
        response.setHeader("Content-Type", "text/plain;charse
        response.setHeader("Cache-Control", "no-cache, no-sto
        response.setHeader("Pragma", "no-cache");
        response.setHeader("my-header", "hello");

        // [Header 편의 메서드]
        // content(response);
        // cookie(response);
        redirect(response);

        // [message body]
        PrintWriter writer = response.getWriter();
        writer.println("ok");
    }
    private void content(HttpServletResponse response) {
        //Content-Type: text/plain;charset=utf-8
        //Content-Length: 2
        //response.setHeader("Content-Type", "text/plain;char
```

```
        response.setContentType("text/plain");
        response.setCharacterEncoding("utf-8");
        //response.setContentLength(2); //(생략시 자동 생성)
    }
    private void cookie(HttpServletResponse response) {
        //Set-Cookie: myCookie=good; Max-Age=600;
        //response.setHeader("Set-Cookie", "myCookie=good; Ma
        Cookie cookie = new Cookie("myCookie", "good");
        cookie.setMaxAge(600); //600초
        response.addCookie(cookie);
    }
    private void redirect(HttpServletResponse response) throw
        //Status Code 302
        //Location: /basic/hello-form.html
        //response.setStatus(HttpServletResponse.SC_FOUND); /
        //response.setHeader("Location", "/basic/hello-form.h
        response.sendRedirect("/basic/hello-form.html");
    }
}
```

HTTP 응답 데이터 - 단순 텍스트, HTML

HTTP 응답 메시지는 주로 다음 내용을 담아서 전달한다.

- 단순 텍스트 응답
  - 앞에서 살펴본 `writer.println("ok").;`
  - HTML 응답
  - HTTP API - MessageBody JSON 응답

HttpServletResponse - HTML 응답

- 자바에서 html 작성... 쉽지 않다.

```
@WebServlet(name = "responseHtmlServlet", urlPatterns = "/res
public class ResponseHtmlServlet extends HttpServlet {
    @Override
    protected void service(HttpServletRequest request, HttpSe
```

```
        // Content-Type: text/html;charset=utf-8
        response.setContentType("text/html");
        response.setCharacterEncoding("utf-8");

        PrintWriter writer = response.getWriter();
        writer.println("<html>");
        writer.println("<body>");
        writer.println(" <div> 안녕? </div>");
        writer.println("</html>");
        writer.println("</body>");
    }
}
```

HTTP 응답 데이터 - API JSON

```
@WebServlet(name = "requestBodyJsonServlet", urlPatterns = "/
public class RequestBodyJsonServlet extends HttpServlet {
    private ObjectMapper objectMapper = new ObjectMapper();
    @Override
    protected void service(HttpServletRequest request, HttpSe
        ServletInputStream inputStream = request.getInputStre
        String messageBody = StreamUtils.copyToString(inputSt

        System.out.println("messageBody = " + messageBody);

        HelloData helloData = objectMapper.readValue(messageB

        System.out.println("helloData.username = " + helloDat
        System.out.println("helloData.age = " + helloData.get
    }
}
```