Spring MVC 02

[서블릿 - 기초]

스프링 MVC 1편 - 백엔드 웹 개발 핵심 기술

김영한

2022.08.18

서블릿 프로젝트 생성

https://start.spring.io/

War는 톰캣 서버를 따로 설치할때 주로 사용하지만 spring web dependencies를 받으면 톰캣이 내장된다.

인텔리제이 커뮤니티 버전에서 war 프로젝트인 경우 인텔리제이의 Build and run using 옵션을 intelliJ IDEA로 변경해 사용할 수 없다. 즉 Gradle로 사용해야 한다.

서블릿

```
@WebServlet(name = "helloServlet", urlPatterns = "/hello")
public class HelloServlet extends HttpServlet {
```

+ 구현 메소드로 요청 메소드를 받는다.

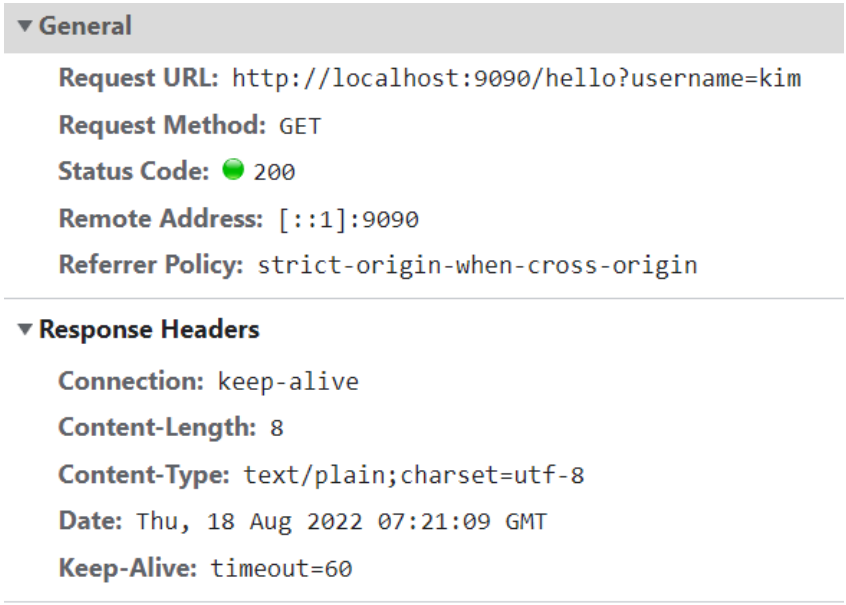
```
String username = request.getParameter("username");

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

service 메소드를 구현해서 사용하고

http://localhost:9090/hello?username=kim

로 요청을 하게 되면 request, response 를 개발자 도구에서 확인이 가능하다.



개발자 도구에서 확인 가능하다.

서블릿 동작 원리

웹 브라우저가 HTTP 요청을 만들어주고 톰캣 서버에서 request 객체를 HTTP 요청으로 만들어주고 request객체가 톰캣 서블릿 컨테이너 내부의 helloServlet 이름을 가진 서블릿 객체를 만들고 실행한다. response 객체를 갱신해 HTTP 요청을 다시 웹 브라우저에 보내준다.

HttpServletRequest

- 서블릿이 HTTP요청 메시지를 파싱해서 편하게 객체에 담아서 제공하는 것

부가 기능

임시 저장소 기능

세션 관리 기능

HTTP 요청 메시지 방법

GET - 쿼리 파라미터

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

user=name,age=20 : 쿼리 파라미터

사용 **:** 메시지 바디가 없다. 검색, 필터, 페이징에서 많이 사용

username=김진호&age=20&username=hello2 같은 이름이 두개인 파라미터?

getParameterValues()로 한꺼번에 조회가 가능하다.

POST - HTML Form 데이터 전송

메시지 바디에 쿼리파라미터 형식으로 전달

x-www-form-urlencoded 형식

user=name,age=20 쿼리 파라미터 (메시지 바디에 존재)

사용 **:** 주로 회원가입, 상품 주문, HTML Form 사용

GET형식이랑 파라미터 형식이 같기 때문에 사용하는 Servlet에서 방법이 똑같다

HTTP message body 에 데이터를 직접 담아서 요청

사용 **:** HTTP API 주로 사용 **JSON**, XML, TEXT

POST, PUT, PATCH에서 주로 사용된다.

message body - TEXT 전송

```
ServletInputStream inputStream = request.getInputStream();
String messageBody = StreamUtils.copyToString(inputStream, StandardCharsets.UTF_8);
```

스프링 유틸리티로 TEXT를 보냈을시에 바로 꺼낼수 있다.

message body - JSON 형식 전송

- 대부분 객체 형식으로 전달을 받아서 파싱해서 사용한다.

message : {"username": "hello", "age": 20}

Text로도 받아서 확인 할 수 있지만 Jackson라이브러리를 사용해서 파싱 할 수 있다.

```
ObjectMapper objectMapper = new ObjectMapper();
ServletInputStream inputStream = request.getInputStream();
String messageBody = StreamUtils.copyToString(inputStream, StandardCharsets.UTF_8);
HelloData helloData = objectMapper.readValue(messageBody, HelloData.class);
```

HttpServletResponse - 기본 사용법

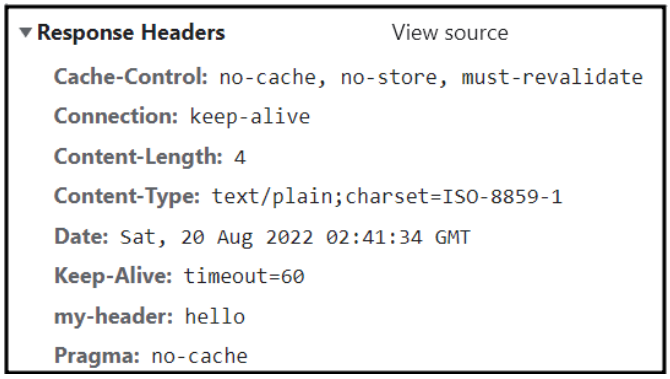
```
response.setStatus(HttpServletResponse.SC_OK);
```

Status line 설정

```
response.setHeader("Content-Type", "text/plain");
response.setHeader("Cache-Control", "no-cache, no-store, must-revalidate");
response.setHeader("Pragma", "no-cache"); → 캐시 무효화
```

```
response.setHeader("my-header", "hello");
```

확인 Response



쿠키 세팅

```
private void cookie(HttpServletResponse response) {
    Cookie cookie = new Cookie("myCookie", "good");
    cookie.setMaxAge(600); //600초
    response.addCookie(cookie);
}
```

redirect 사용 주석은 하나씩 세팅 , 마지막은 redirect 메소드로 직접사용

```
private void redirect(HttpServletResponse response) throws IOException {
    //Status Code 302
    //Location: /basic/hello-form.html
    //response.setStatus(HttpServletResponse.SC_FOUND); //302
    //response.setHeader("Location", "/basic/hello-form.html");
    response.sendRedirect("/basic/hello-form.html");
}
```

응답 데이터 **TEXT, HTML, HTTP API**

text

⇒ response.getWriter()print("ok");

HTML

```
response.setContentType("text/html");
response.setCharacterEncoding("utf-8");
```

를 설정한 후에

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

html을 그대로 작성해 주면 동작한다.

JSON 형식

`application/json content형식을 지정해 준다.`

```
response.setContentType("application/json");
response.setCharacterEncoding("utf-8");

HelloData helloData = new HelloData();
helloData.setUsername("kim");
helloData.setAge(20);

String result = objectMapper.writeValueAsString(helloData);
response.getWriter().write(result);
```