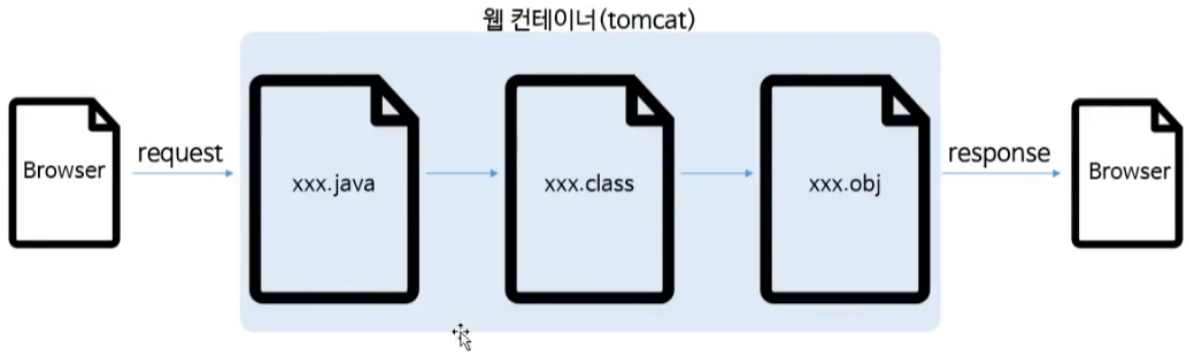
JSP & Servlet - Servlet(기본)

💡 서블릿의 개념을 이해 하자 (중요 !!!)

이전까지 배웠던 방식은 .jsp 파일 사용해서 동적인 데이터를 만들어 내는 방법을 학습 했다.
이번에는 통칭 servlet 이라는 순수 자바 코드 만을 작성해서 요청 및 응답을 처리하는 코딩을 배워 보자

서블릿 으로 만들었을 대 WAS 동작을 확인해 보자.

> .jsp 파일과 다른 부분을 꼭 확인하세요



서블릿 파일 생성하기

> 서블릿에 접근할려면 무엇을 먼저 해야 될까
> URL Mapping 을 이해 하자.
> 여기서는 @어노테이션을 이용해서 url mapping 를 사용

MyServlet.java 파일 생성

```
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;


@WebServlet("/ms1")
public class MyServlet extends HttpServlet {
  private static final long serialVersionUID = 1L;


    public MyServlet() {
        super();
    }

  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    response.getWriter().append("Served at: ").append(request.getContextPath());
    System.out.println("여기는 get 방식으로 접근했어요!!! ");
  }


  protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
     // doGet(request, response);
    System.out.println("여기는 post 방식으로 접근했어요!!! ");
  }

}
```

☐ 문제

OutStreamWrite 이용해서 응답 처리 : CSS 도 사용해 보자.

> 적용이 안될 경우 서버를 재 시작해야 합니다.

```
@WebServlet("/ms1")
public class MyServlet extends HttpServlet {
  private static final long serialVersionUID = 1L;

  public MyServlet() {
    super();
  }
  // request, response 객체는 웹 컨테이너가 먼저 만들어 주는 녀셕이다.
  // get 요청시 콜백 반응
  protected void doGet(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException {
    PrintWriter out = response.getWriter();
    out.print("<html>");
    out.print("<head>");
    out.print("</head>");
    out.print("<body>");
    out.print("<section>");
    out.print("<p style=\"color: red\">");
```

```
    out.print("Hello Fisrt Servlet ~ by boot ");
    out.print("</p>");
    out.print("</section>");
    out.print("</body>");
    out.print("</html>");
  }

  protected void doPost(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException {
    // doGet(request, response);
    System.out.println("여기는 post 방식으로 접근했어요!!! ");
  }

}
```

☐ 문제

> WAS 의 동작 방식에 대한 이해
> JSON 형식으로 응답 처리
> request 객체와 response 객체를 살펴 보고 할 수 있는 것들을 확인해 보자.
> 한글 깨짐 문제를 확인하고 해결 방안 모색
> MIME TYPE 에 대한 이해

```
1. [xxx.java](http://xxx.java) 파일과 xxx.class 파일이 어디에 생성 되었는지 확인하고 경로를 적어 주세요

2. 순수 자바코드인 서블릿을 이용해서 json 형식으로 데이터를 리턴해 보자.
ex)
{
"name": “홍길동",
"age": 10,
"isMarried": false
}

3. 한글이 만약 깨진다면 해결 하시오
4. MIME TYPE 을 설정해서 응답 결과를 그냥 평문으로 리턴해 보자.
```

https://developer.mozilla.org/ko/docs/Web/HTTP/Basics_of_HTTP/MIME_types

해설

```
@WebServlet("/ms2")
public class MyServlet2 extends HttpServlet {
  private static final long serialVersionUID = 1L;

  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

    // response 객체에 MIME TYPE 을 지정하자
    response.setContentType("application/json; charset=utf-8");
    // response.setContentType("text/plain; charset=utf-8");
```

```
    // JSON 형식도 결과적으로 문자열 이다 . 패턴이 있는 문자열
    String resJson ="{\r\n"
        + "\"name\": \"홍길동\",\r\n"
        + "\"age\": 10,\r\n"
        + "\"isMarried\": false\r\n"
        + "}";

    // 응답 객체에 담아야 사용자가 받아 볼 수 있다 --> response 이다.
    response.getWriter().write(resJson);

  }

  protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    doGet(request, response);
  }
}
```

Http 요청과 응답에 대한 HTTP 메세지 예시