



2025년 상반기 K-디지털 트레이닝

JSP의 이해

[KB] IT's Your Life

✓ JSP(Java Server Page)

- 태그 기반의 웹 컴포넌트로서 jsp 확장자를 가짐
- 클라이언트의 요청에 의해 동적으로 실행
- 자동으로 서블릿으로 변환되어 실행 
- MVC 패턴의 View 역할 

html 출력전용 역할

✓ JSP 동작 3단계

○ 변환 단계 ✓

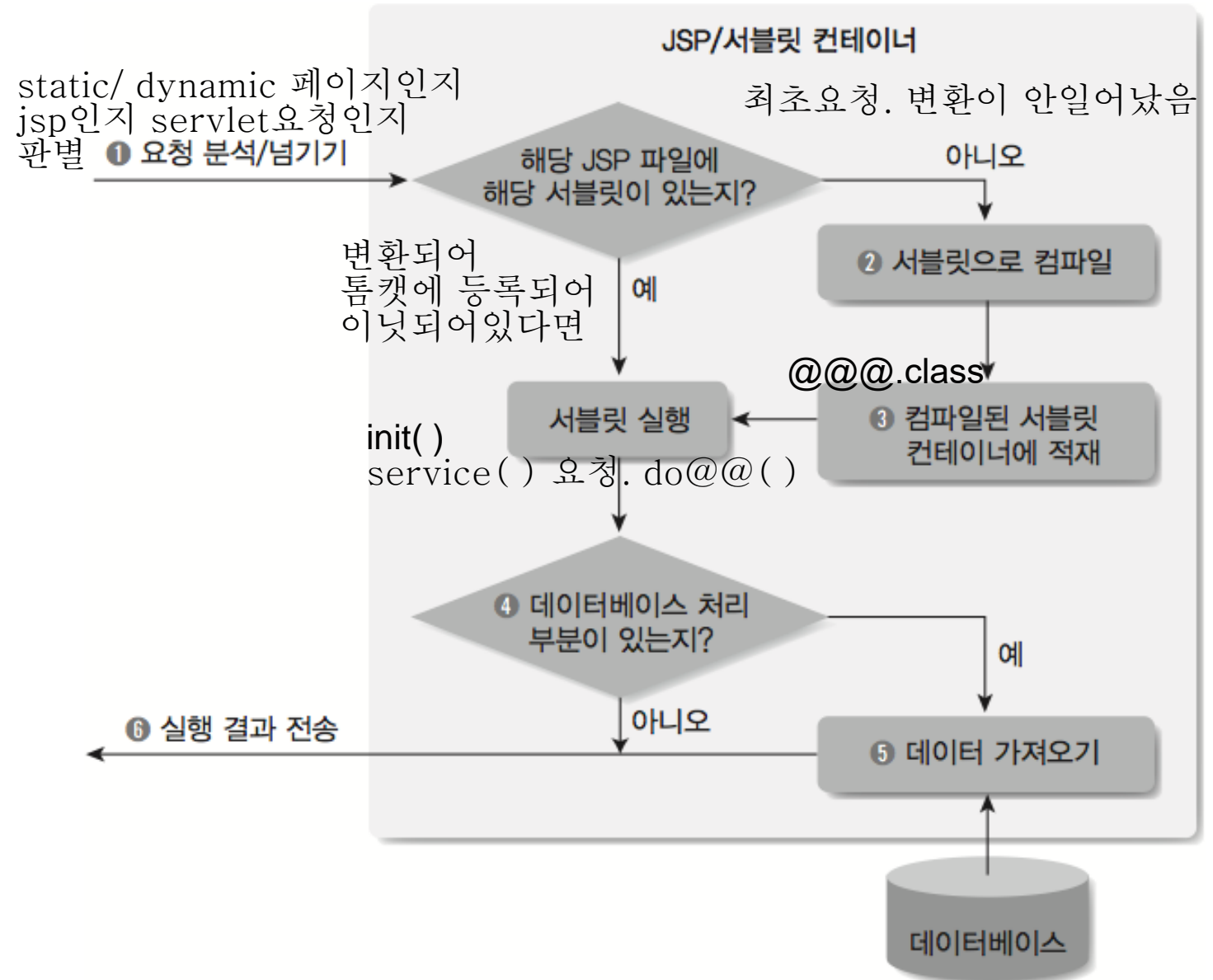
- 요청된 JSP 파일은
파일명_jsp.java 파일명을 가진
서블릿으로 변환

○ 컴파일 단계

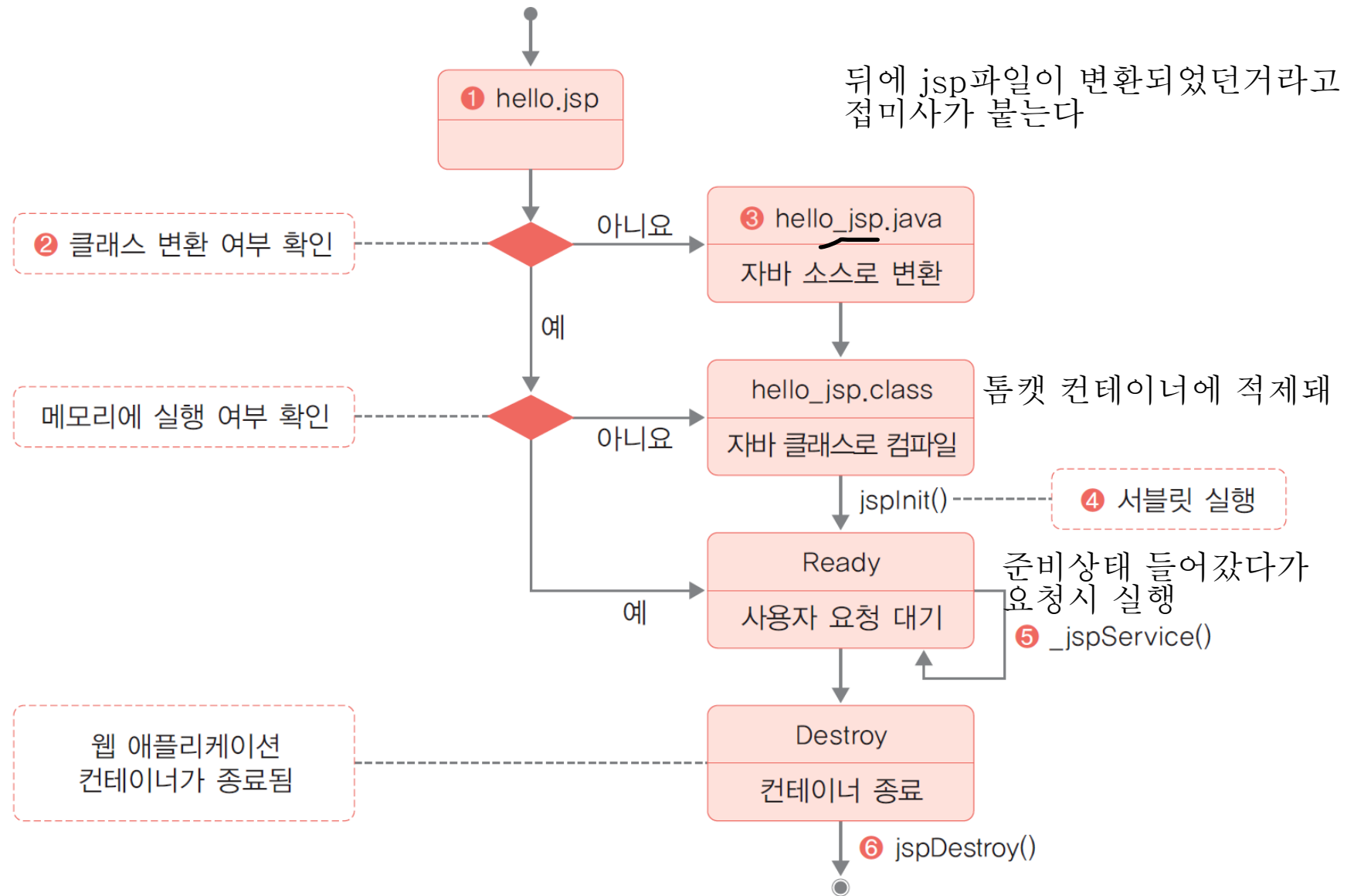
- 변환된 서블릿을 컴파일하는 단계
- 컴파일 에러가 발생하면 변환되지 않음
- 파일명_jsp.class 형식의 파일 생성

○ 실행 단계

- 컴파일 된 파일명_jsp.class 파일을 실행



✓ JSP 동작 과정



✓ JSP 변환 파일 위치

○ 개발시

- C:\Users\<userid>\AppData\Local\JetBrains\IntelliJdea2024.3\tomcat\<임시코드>\work\Catalina\localhost\ROOT\org\apache\jsp\패키지명
- index.jsp → index_jsp.java → index_jsp.class
jsp 서블릿

○ 운영시

- 톰캣홈디렉토리\work\Catalina\localhost\ROOT\org\apache\jsp\패키지명

둘이 비슷하죠?

.java와 .class 파일이 있다

work 폴더의 역할

jsp파일을 변환해서
보관하는 곳

index_jsp.java

```
package org.apache.jsp;

import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.jsp.*;

public final class index_jsp extends org.apache.jasper.runtime.HttpJspBase
    implements org.apache.jasper.runtime.JspSourceDependent,
               org.apache.jasper.runtime.JspSourceImports {
    ...

    public void _jspInit() {
}

    public void _jspDestroy() {
}
```

index_jsp.java

```
public void _jspService(final javax.servlet.http.HttpServletRequest request, final
javax.servlet.http.HttpServletResponse response)
    throws java.io.IOException, javax.servlet.ServletException {

    if (!javax.servlet.DispatcherType.ERROR.equals(request.getDispatcherType())) {
        final java.lang.String _jspx_method = request.getMethod();
        if ("OPTIONS".equals(_jspx_method)) {
            response.setHeader("Allow","GET, HEAD, POST, OPTIONS");
            return;
        }
        if (!"GET".equals(_jspx_method) && !"POST".equals(_jspx_method) && !"HEAD".equals(_jspx_method)) {
            response.setHeader("Allow","GET, HEAD, POST, OPTIONS");
            response.sendError(HttpServletResponse.SC_METHOD_NOT_ALLOWED, "JSP들은 오직 GET, POST 또는 HEAD 메소드만을 허용합
니다. Jasper는 OPTIONS 메소드 또한 허용합니다.");
            return;
        }
    }
}
```

index_jsp.java

```
try {
    response.setContentType("text/html; charset=UTF-8");
    pageContext = _jspxFactory.getPageContext(this, request, response, null, true, 8192, true);
    ...
    out = pageContext.getOut();
    _jspx_out = out;
    out.write("\n");
    out.write("<!DOCTYPE html>\n");
    out.write("<html>\n");
    out.write("<head>\n");
    out.write("    <title>JSP - Hello World</title>\n");
    out.write("</head>\n");
    out.write("<body>\n");
    out.write("<h1>");
    out.print( "Hello World!" );
    out.write("\n");
    out.write("</h1>\n");
    out.write("<br/>\n");
    out.write("<a href=\"hello-servlet\">Hello Servlet</a>\n");
    out.write("</body>\n");
    out.write("</html>");
}
```

준비안한 지역변수들을 사용○하는 것을 볼 수 있다.
pageContext, application, 등등등
jsp에서 기본적으로 제공되는 객체들이다.
지역변수, 객체, 멤버 일 수도 있다.

index.jsp

프로젝트의 webapp디렉 밑에 있다

```
<%@ page contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8" %>
<!DOCTYPE html>
<html>
<head>
    <title>JSP - Hello World</title>
</head>
<body>
<h1><%= "Hello World!" %>
</h1>
<br/>
<a href="hello-servlet">Hello Servlet</a>
</body>
</html>
```



☑ 서블릿과 JSP 비교

	서블릿	JSP
형식	<div> <div>Java 코드</div> <div>HTML 코드</div> </div>	<div> <div>HTML 코드</div> <div>Java 코드</div> </div>
특징	Java 코드 내에 HTML 코드가 삽입되는 형태이다. 따라서 HTML를 작성하는 Java 코드 작업이 복잡하다.	Java 코드 내에 Java 코드가 삽입되는 형태이다. 서블릿에 비해서 HTML 코드를 쉽게 작성할 수 있다.
목적	Java 코드를 이용한 Business Logic 처리에 적합하다. 따라서 MVC 패턴의 Controller 역할로 사용된다.	tag를 이용한 Presentation Logic 처리에 적합하다. 따라서 MVC 패턴의 View 역할로 사용된다.