Chapter 17

디렉터리 패턴

서블릿 패턴 매핑 방법

서블릿 매핑

■ 서블릿 매핑(mapping)

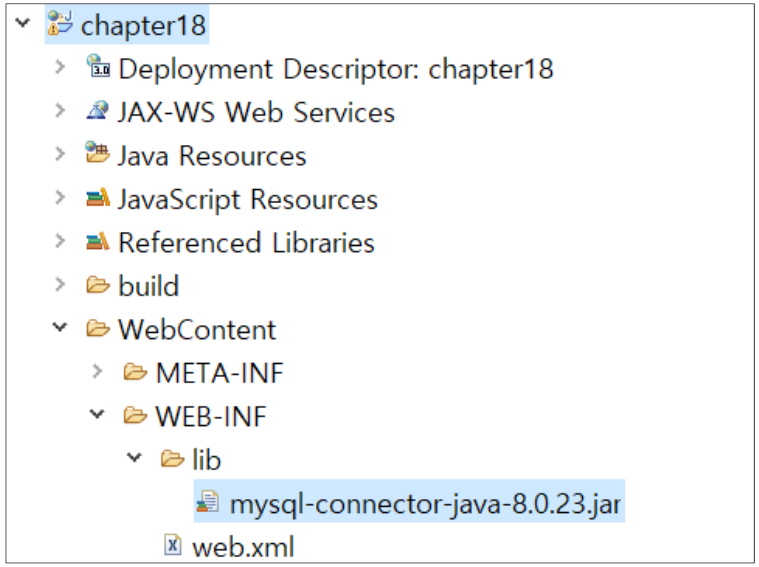
- **서블릿의 접속 경로가 브라우저의 주소표시줄에 노출되면 보안상 문제가 발생될 수 있음**
  - 매핑을 통해 경로를 감춰 보안을 유지할 수 있음
  - 긴 서블릿 이름을 간결하게 표현할 수 있음
  - Web.xml 파일을 사용하거나 애너테이션을 사용해 지정할 수 있음

- **애너테이션을 사용하는 경우 서블릿 매핑은 다양한 방법으로 지정할 수 있음**
  - 애너테이션에 패턴의 이름을 지정하는 방법
    - 지금까지 실습했던 방법
  - 애너테이션의 패턴으로 디렉터리를 사용하는 방법
    - 실제 디렉터리를 생성하고 문서를 저장하는 것이 아니라, 디렉터리의 이름을 패턴으로 지정하는 방법
  - 애너테이션의 패턴으로 확장자를 사용하는 방법
    - 일반적으로 do 확장자를 지정해 사용하고 있음

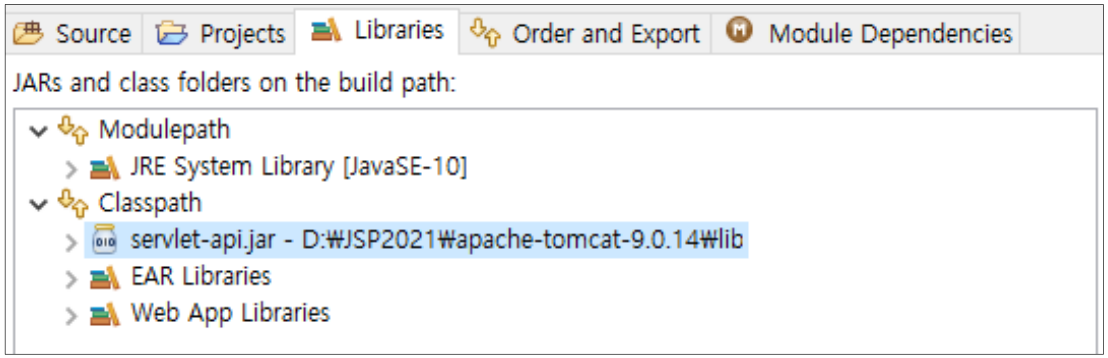
서블릿 매핑

■ 프로젝트 생성

- 프로젝트 이름 : chapter18
  - web.xml 파일은 반드시 생성해야 함
  - Connector 추가
    - 이전 프로젝트에서 복사 가능



- servlet-api 라이브러리 추가



서블릿 매핑

Mapping_PatternName.java

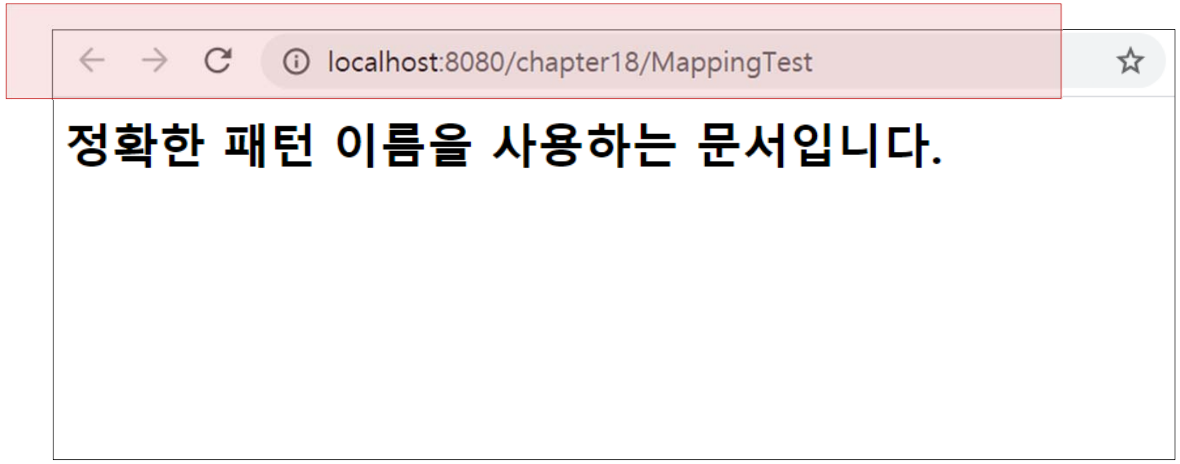
```java
package chapter18;

import java.io.IOException;
import java.io.PrintWriter;

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("/MappingTest")
public class Mapping_PatternName extends HttpServlet {
    private static final long serialVersionUID = 1L;

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

        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();

        out.println("<html><body>");
        out.println("<h2>정확한 패턴 이름을 사용하는 문서입니다.</h2>");
        out.println("</body></html>");
    }

}
```

서블릿 매핑

■ 애너테이션에 패턴의 이름을 지정한 경우

- URL에 정확한 패턴의 이름을 입력해야 해당 서블릿 문서를 실행할 수 있음

http://localhost:8080/chapter18/MappingTest



서블릿 매핑(mapping)

■ 이전 실습에서 컨트롤러를 사용하는 방법

- 이전 실습에서 컨트롤러에 액션 코드를 전달해 DAO의 메서드를 호출했었음
  - 컨트롤러에 액션 코드를 전달하기 위해서는 <input>요소의 hidden 속성이나 쿼리 스트링을 사용하였음

```
<form name="memberInsert" method=post action="MemberController.jsp">
    <input type=hidden name=action value="insert">
```

```
response.sendRedirect("MemberController.jsp?action=list");
```

  - 컨트롤러에 액션 코드의 값을 가지는 추가적인 파라미터를 URL과 함께 전달 하였음

- 애너테이션의 패턴으로 디렉터리나 확장자를 지정할 경우
  - URL 만으로 컨트롤러에 매핑할 수 있음

디렉터리 패턴

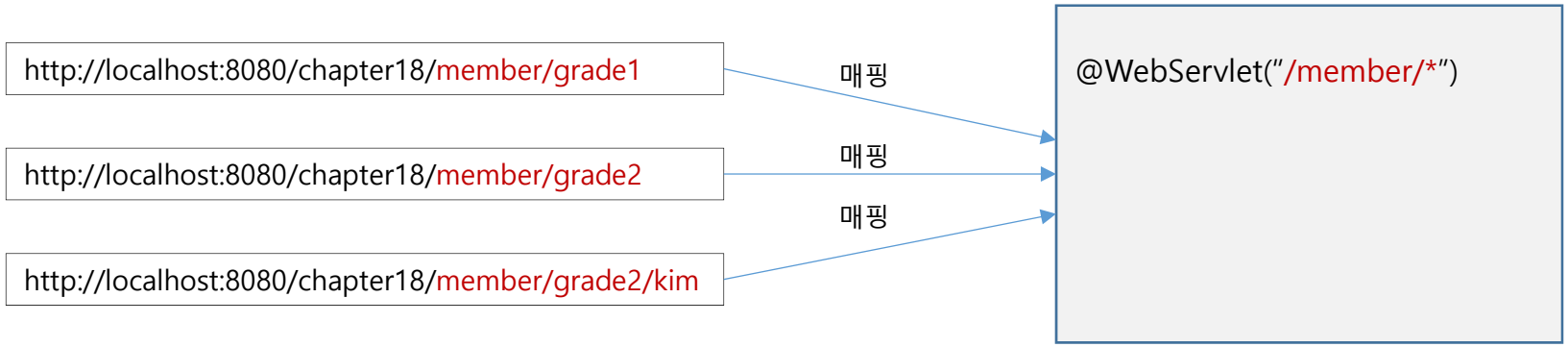
■ 디렉터리 패턴

- **애너테이션의 패턴으로 디렉터리를 지정하는 방법**

```
@WebServlet("/디렉터리/*")
```

- 문서를 지정된 디렉터리에 저장하는 것이 아니라 애너테이션으로 디렉터리 이름을 사용하는 것임
- 디렉터리와 함께 정확인 이름을 지정하는 것이 아니라 *를 사용해 패턴으로 지정함
  - URL에 디렉터리를 포함하고 있는 모는 것이 서블릿에 매핑된다는 것을 의미함
- 반드시 /로 시작해야 함

- 매핑의 예
  - 어플리케이션의 이름이 chapter18인 경우



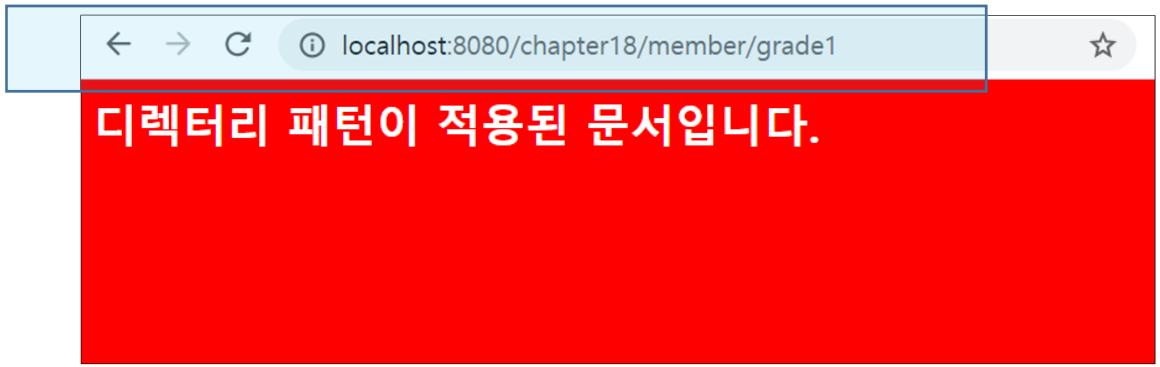
디렉터리 패턴

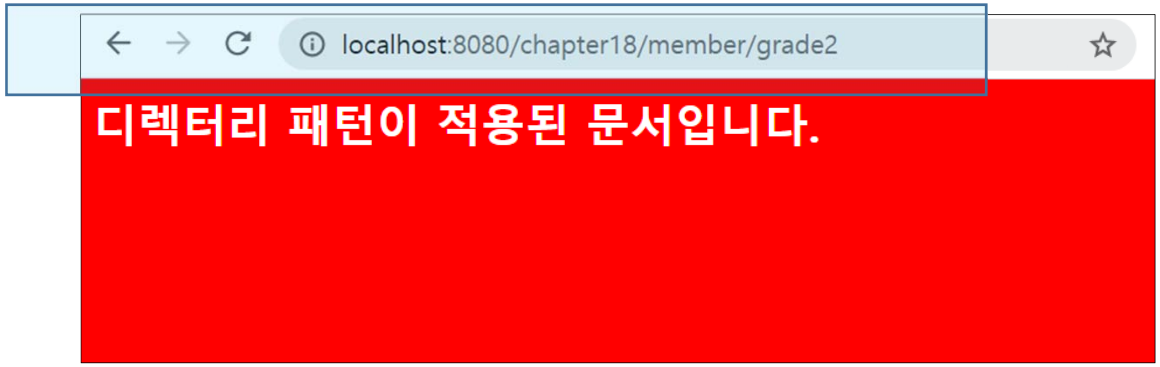
Mapping_Dir.java

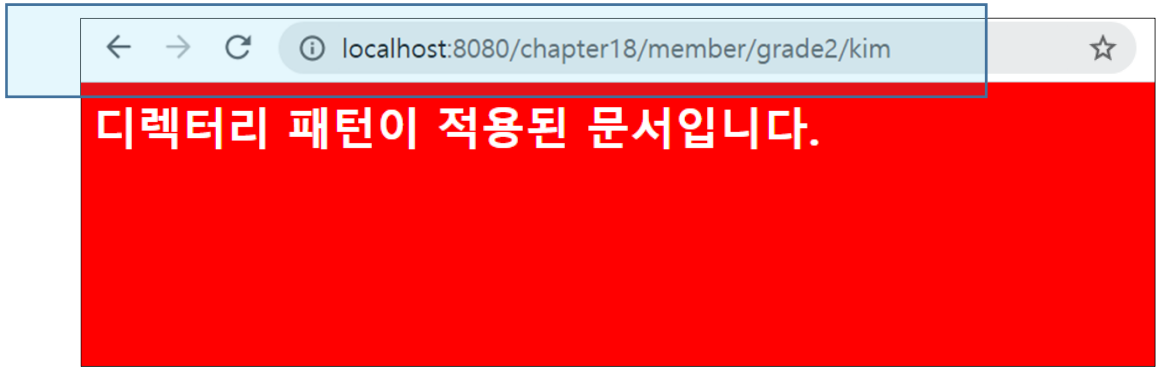
```java
package chapter18;

import java.io.IOException;
import java.io.PrintWriter;

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("/member/*")
public class Mapping_Dir extends HttpServlet {
    private static final long serialVersionUID = 1L;

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

        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();

        out.println("<html><body bgcolor='red' text='white'>");
        out.println("<h2>디렉터리 패턴이 적용된 문서입니다.</h2>");
        out.println("</body></html>");

    }

}
```

디렉터리 패턴

Mapping_Dir.java

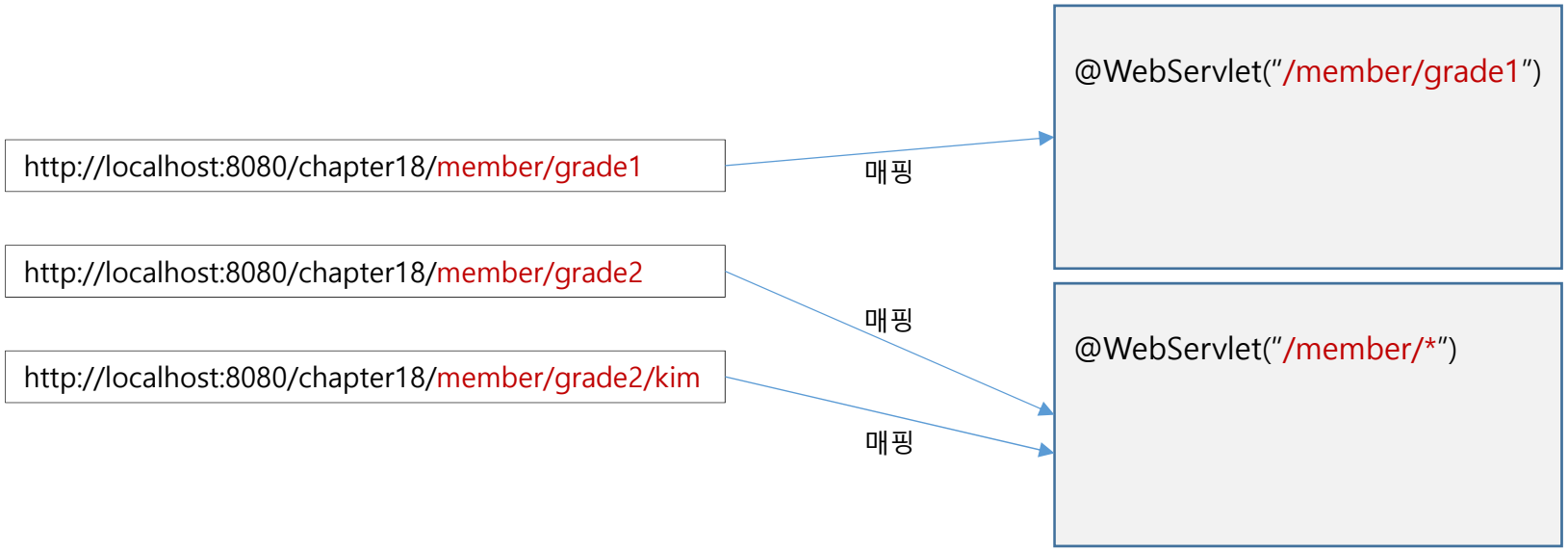






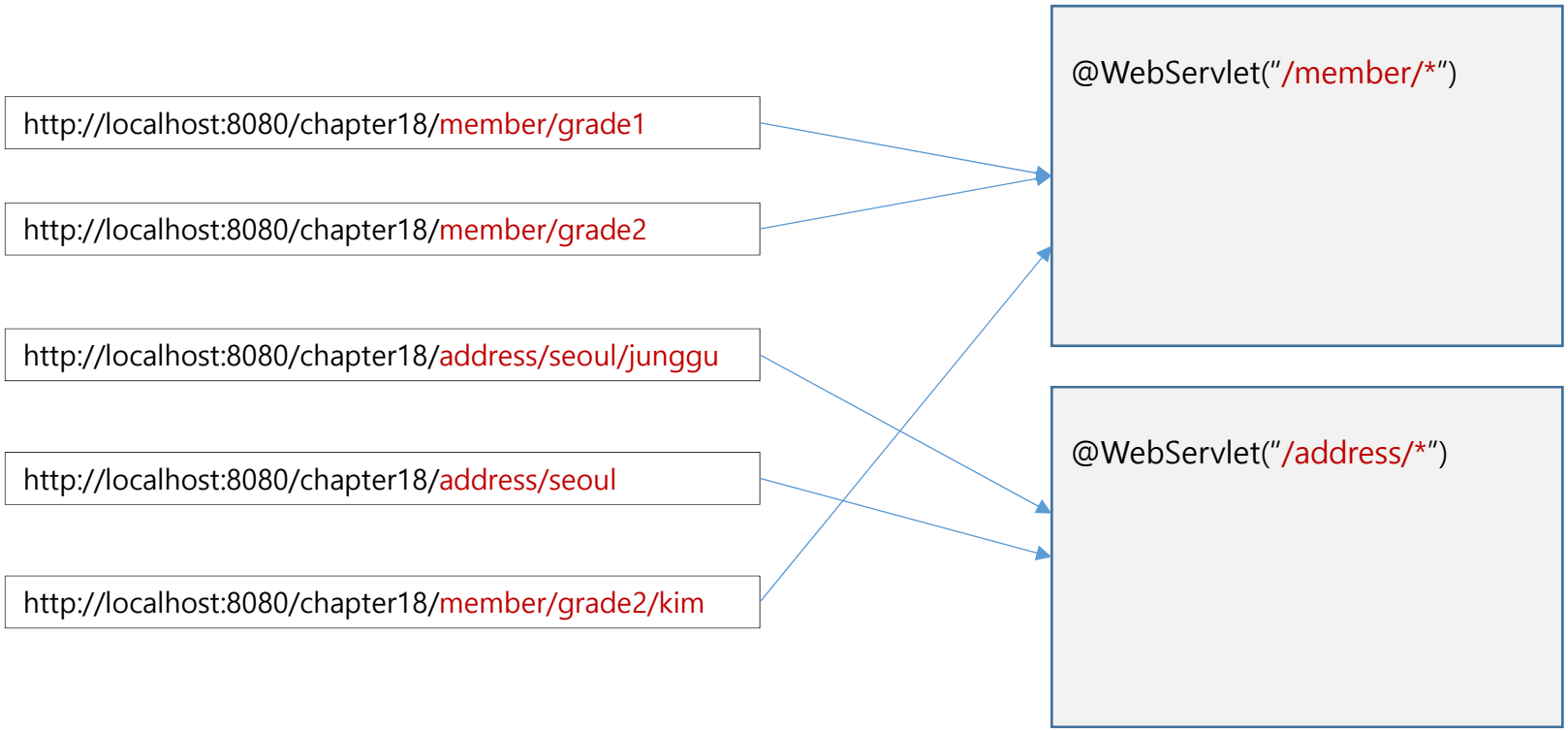
디렉터리 패턴

- 애너테이션의 패턴 이름으로 * 문자가 아닌 이름을 지정할 경우
  - 지정한 이름이 포함된 URL이 있다면, 그 URL은 이름이 지정된 서블릿으로 매핑됨



디렉터리 패턴

- 디렉터리 패턴을 사용하면 여러 개의 서블릿(컨트롤러)에 URL을 매핑할 수 있음



확장자 패턴

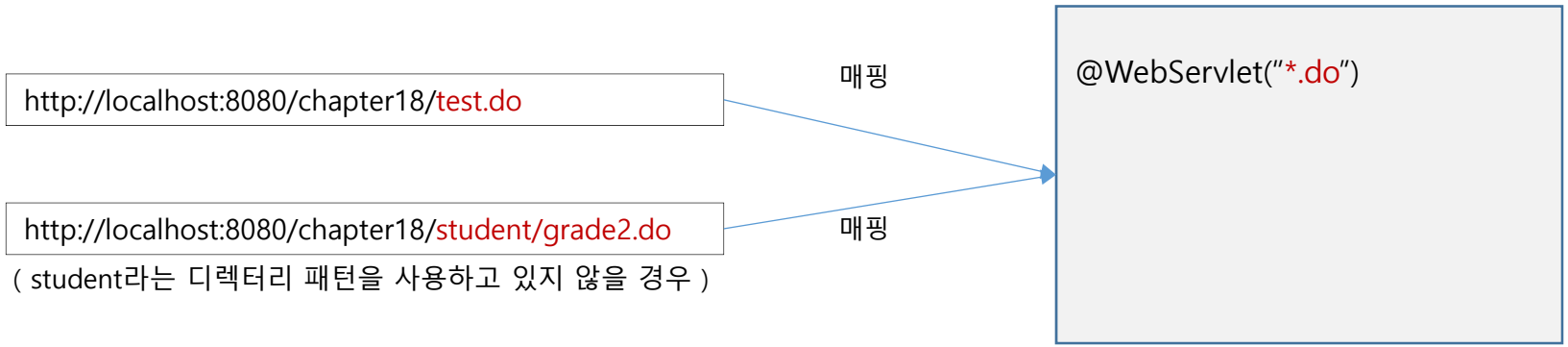
■ 확장자 패턴

- **애너테이션 패턴으로 특정 확장자를 지정하는 방법**

```
@WebServlet("*.확장자*")
```

- 파일의 이름을 정확히 지정하기 않고 * 문자를 사용해 패턴으로 지정
- 패턴의 첫 문자라 / 문자가 아니어야 함

- 확장자 매핑의 예
  - 어플리케이션의 이름이 chapter14인 경우



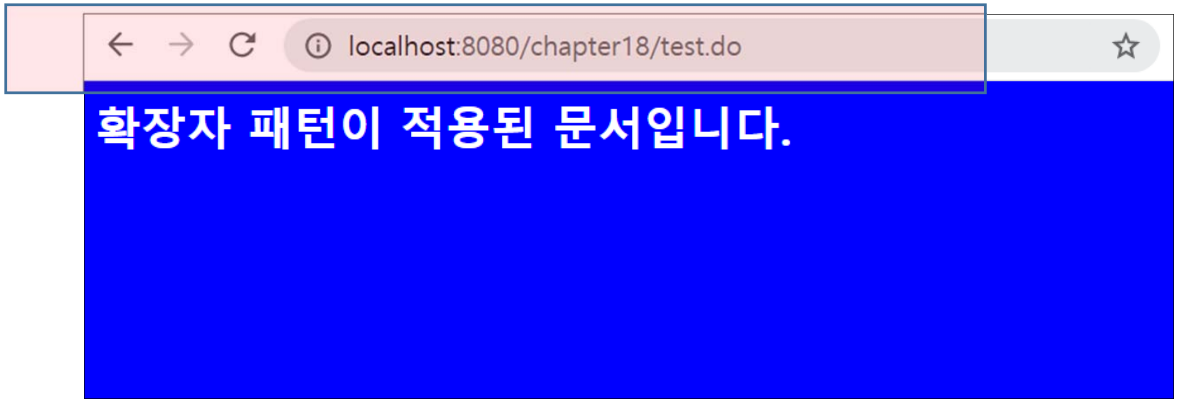
확장자 패턴

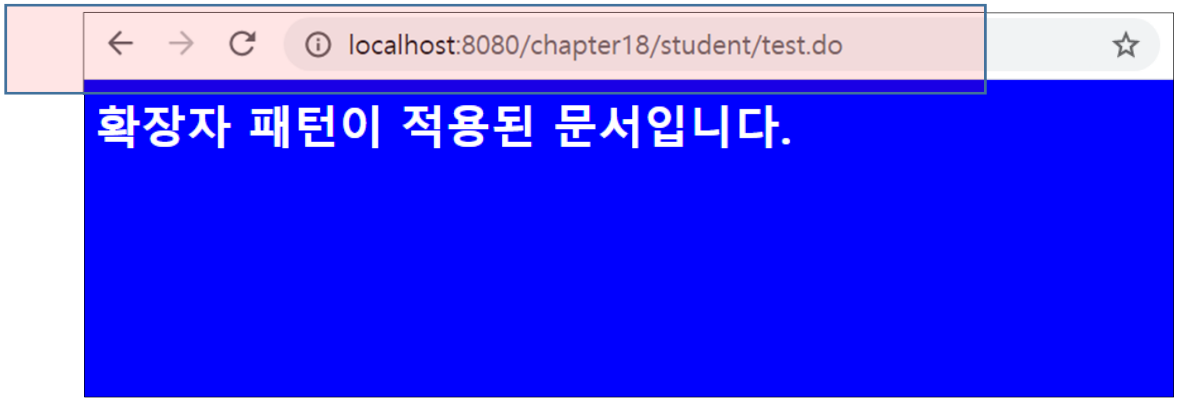
Mapping_Ext.java

```java
package chapter18;

import java.io.IOException;
import java.io.PrintWriter;

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("*.do")
public class Mapping_Ext extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();

        out.println("<html><body bgcolor='blue' text='white'>");
        out.println("<h2>확장자 패턴이 적용된 문서입니다.</h2>");
        out.println("</body></html>");
    }

}
```

확장자 패턴

Mapping_Ext.java



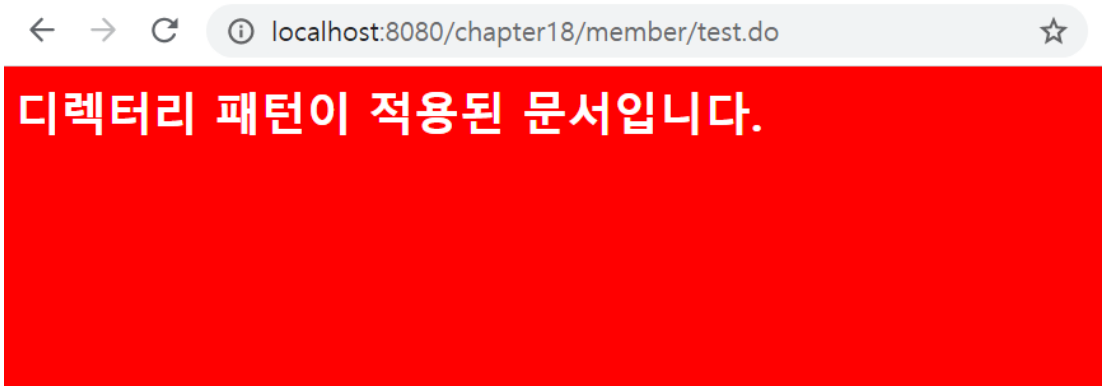


디렉터리 패턴과 확장자 패턴

■ 디렉터리 패턴과 확장자 패턴을 동시에 사용할 경우

- **디렉터리 패턴이 우선함**

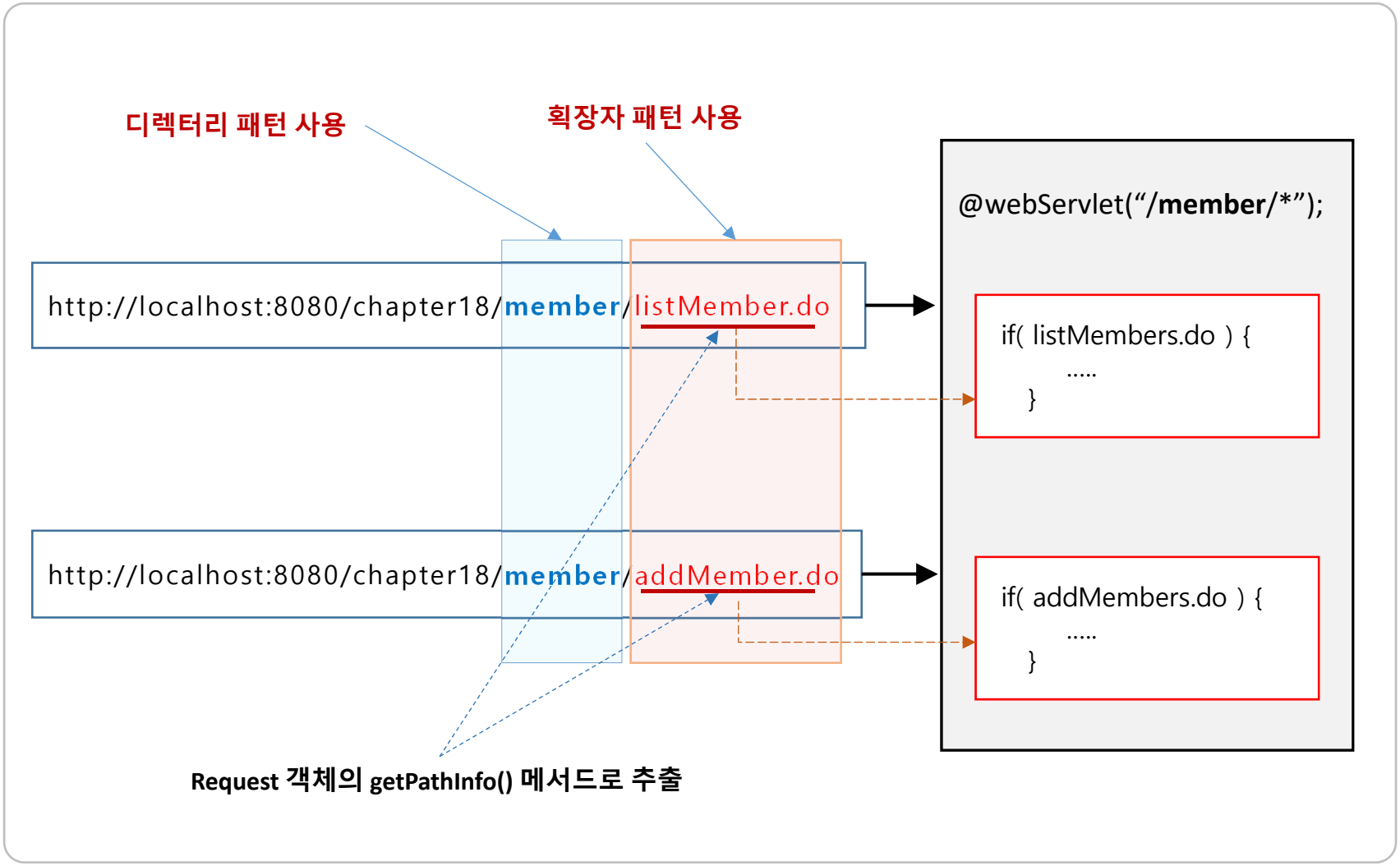
http://localhost:8080/chapter18/member/test.do



- **주로 디렉터리 패턴과 확장자 패턴을 혼용하여 사용함**
  - 디렉터리 패턴으로 컨트롤러에 매핑
  - 확장자 패턴을 액션 코드로 사용
    - URL에 포함되어 있는 확장자 패턴은 getPathInfo( ) 메서드를 사용해 추출할 수 있음

디렉터리 패턴과 확장자 패턴

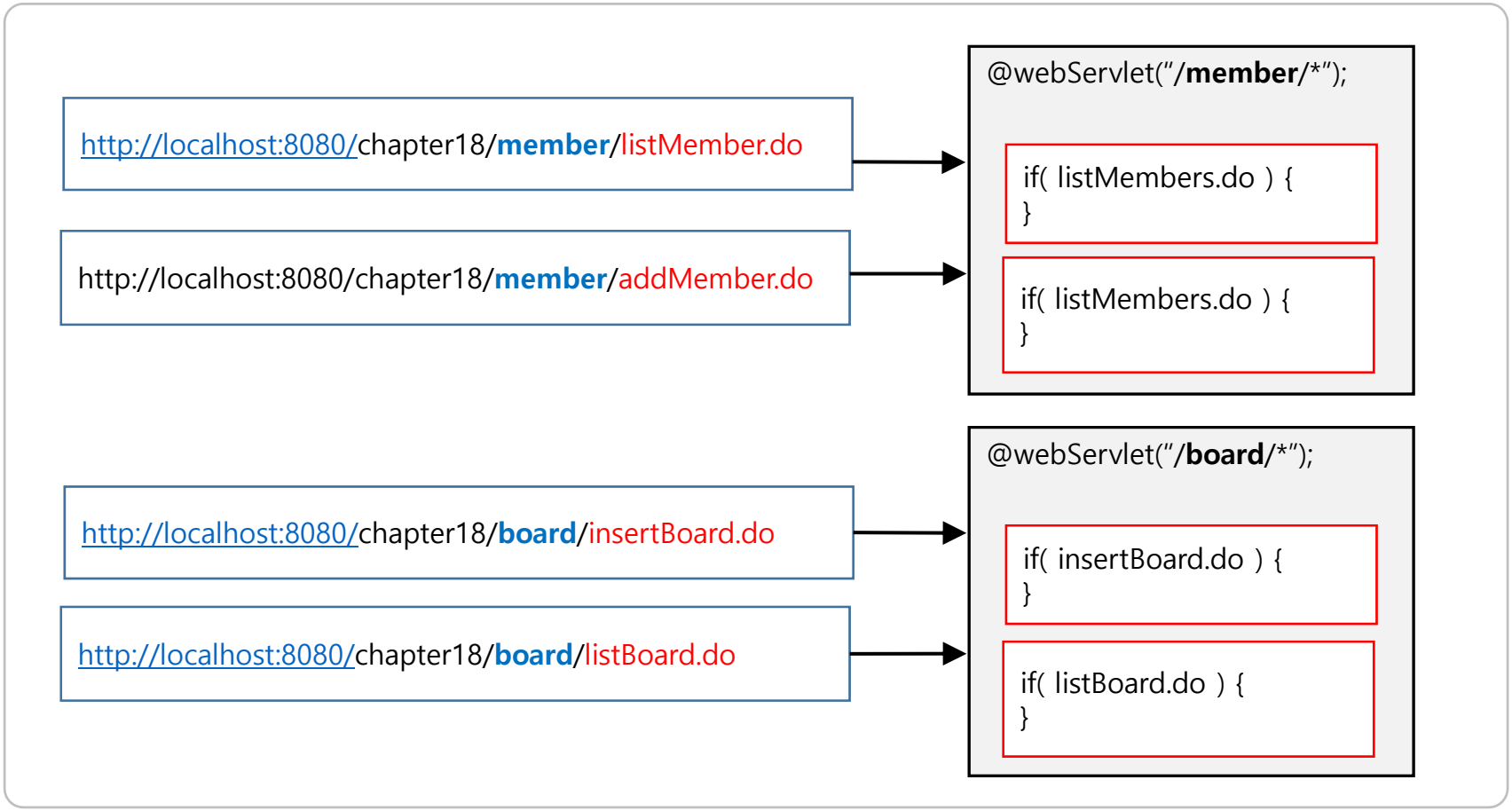
■ 디렉터리 패턴과 확장자 패턴 사용의 예 1



디렉터리 패턴과 확장자 패턴

■ 디렉터리 패턴과 확장자 패턴 사용의 예

- 기능에 따라 여러 개의 컨트롤러를 사용할 수 있음



수고하셨습니다