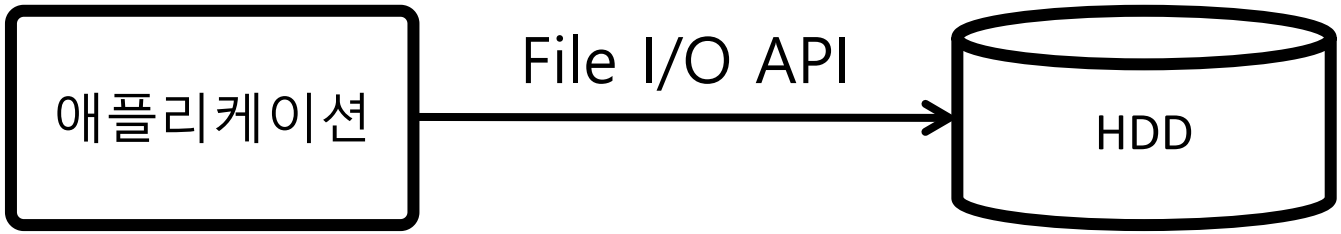
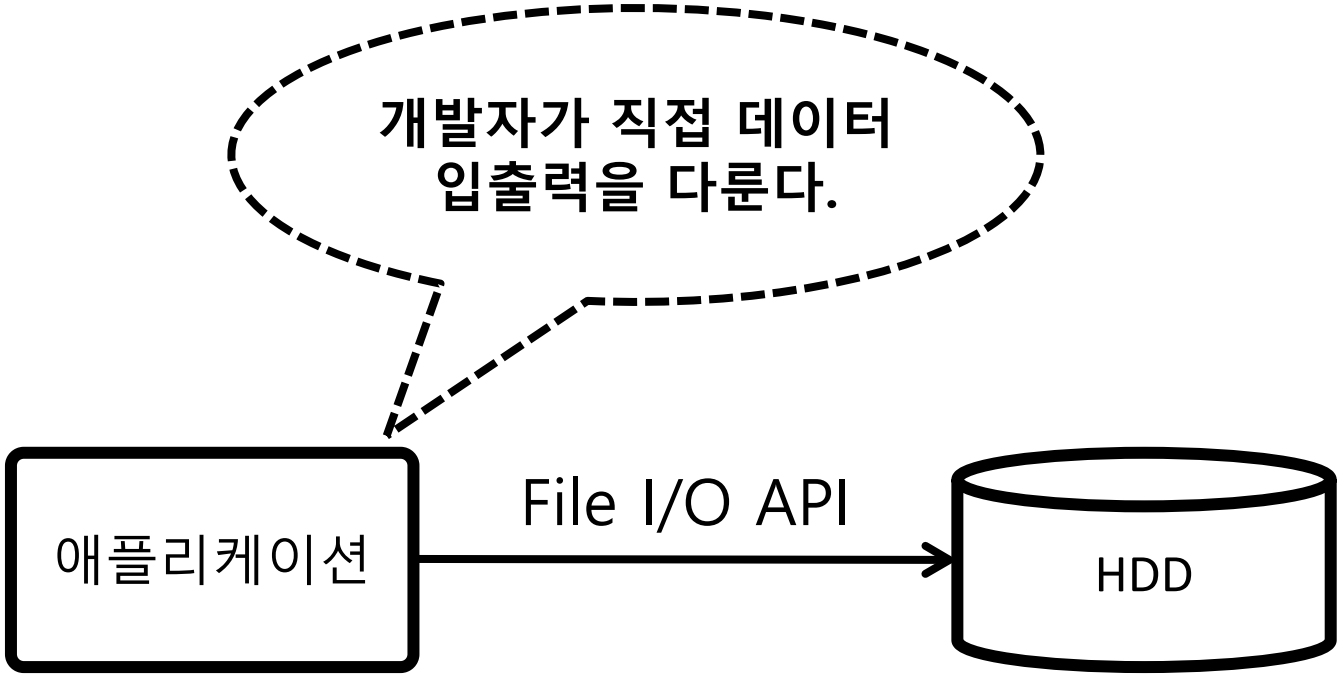
서블릿과 JDBC

DBMS의 등장 전!

DBMS 등장 전

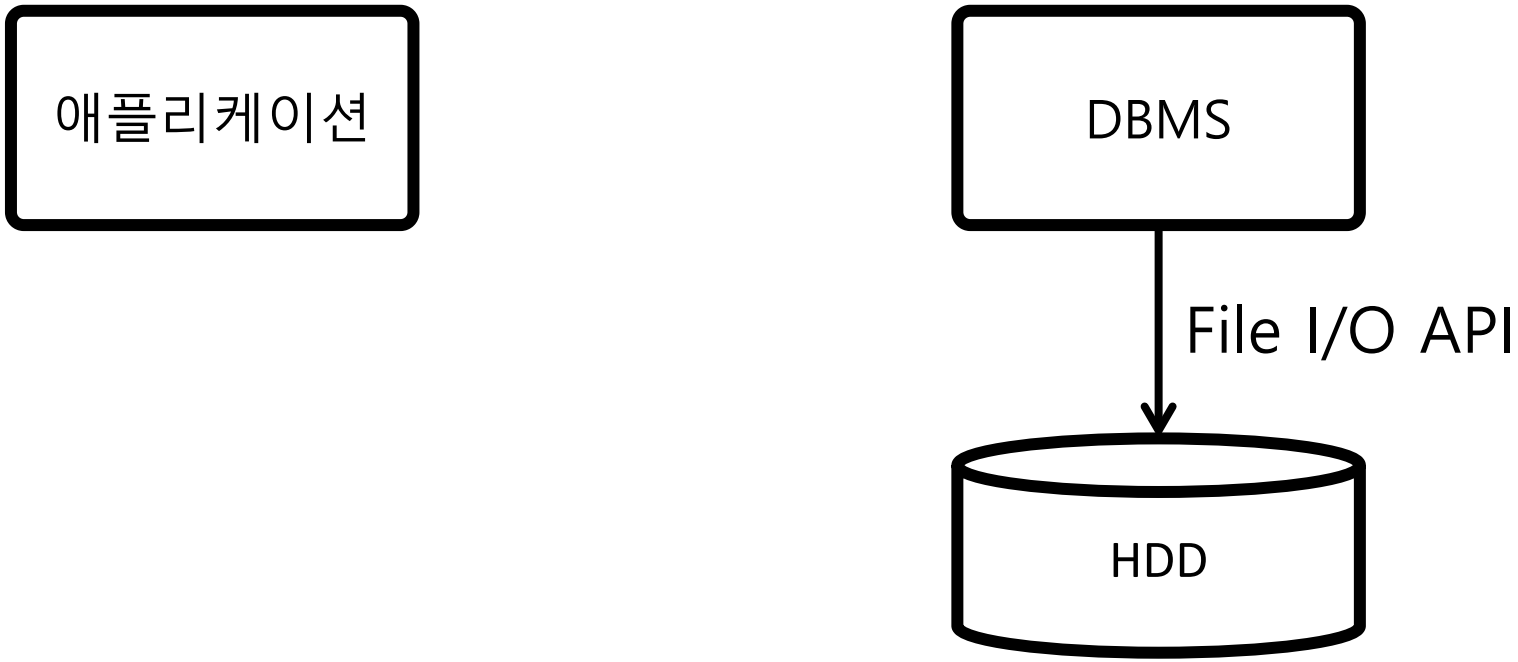


DBMS 등장 전

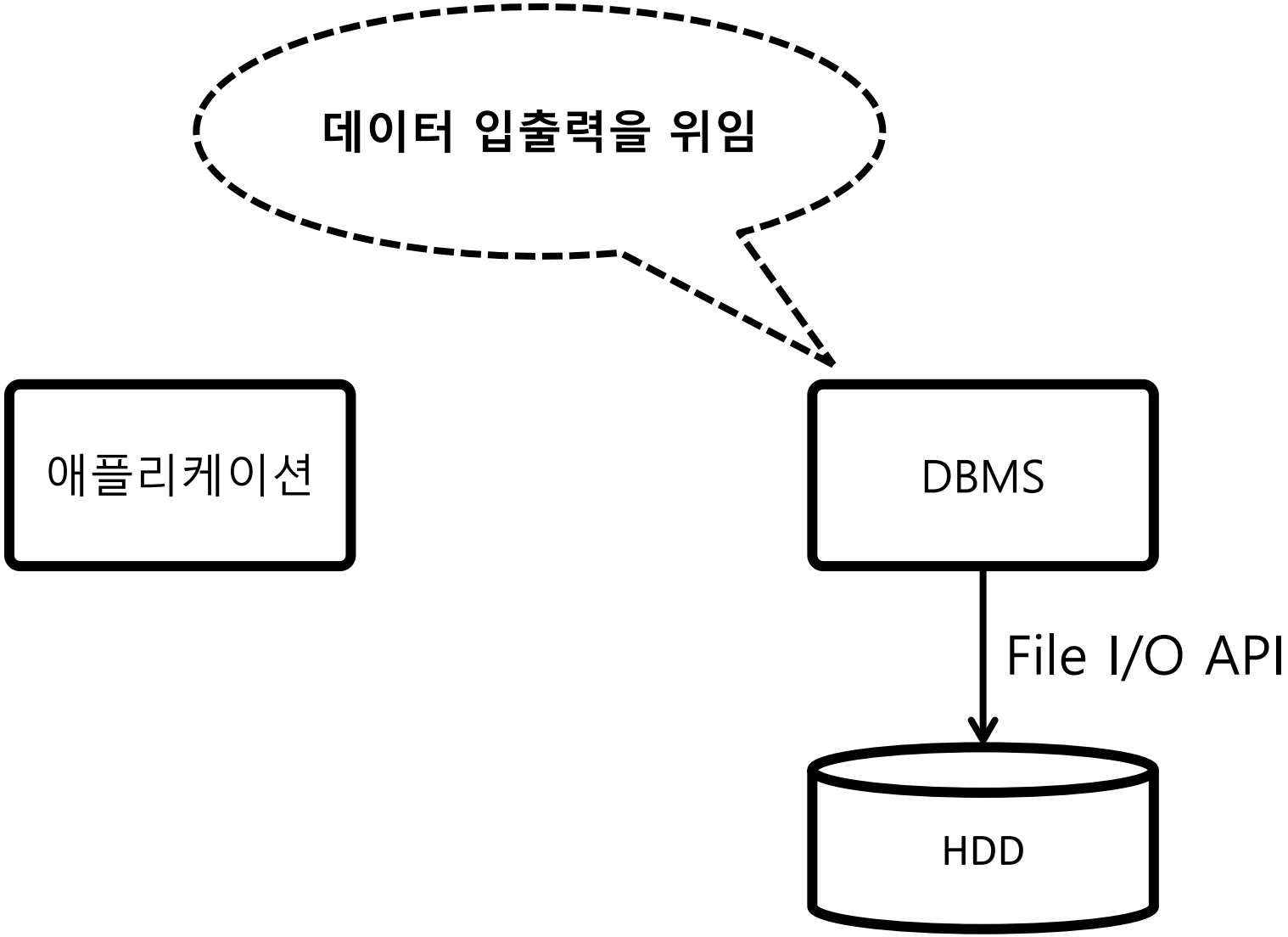


DBMS 등장

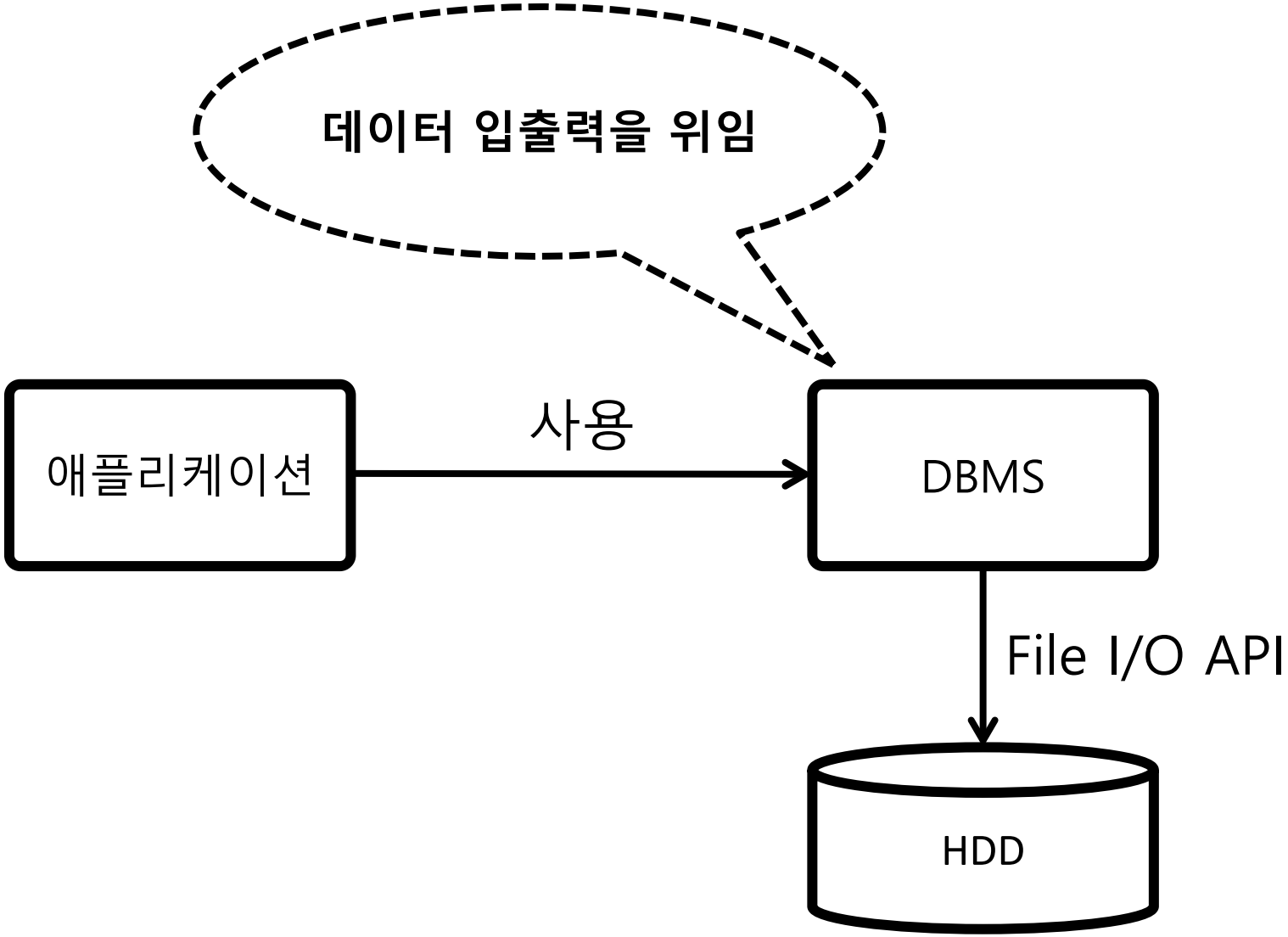
DBMS 도입



DBMS 도입



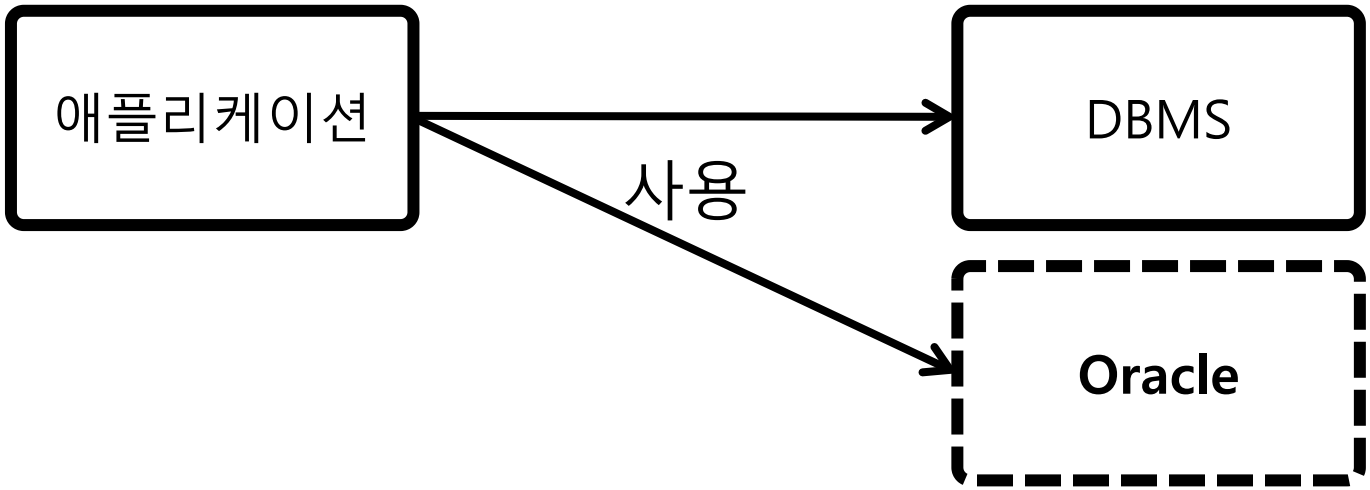
DBMS 도입



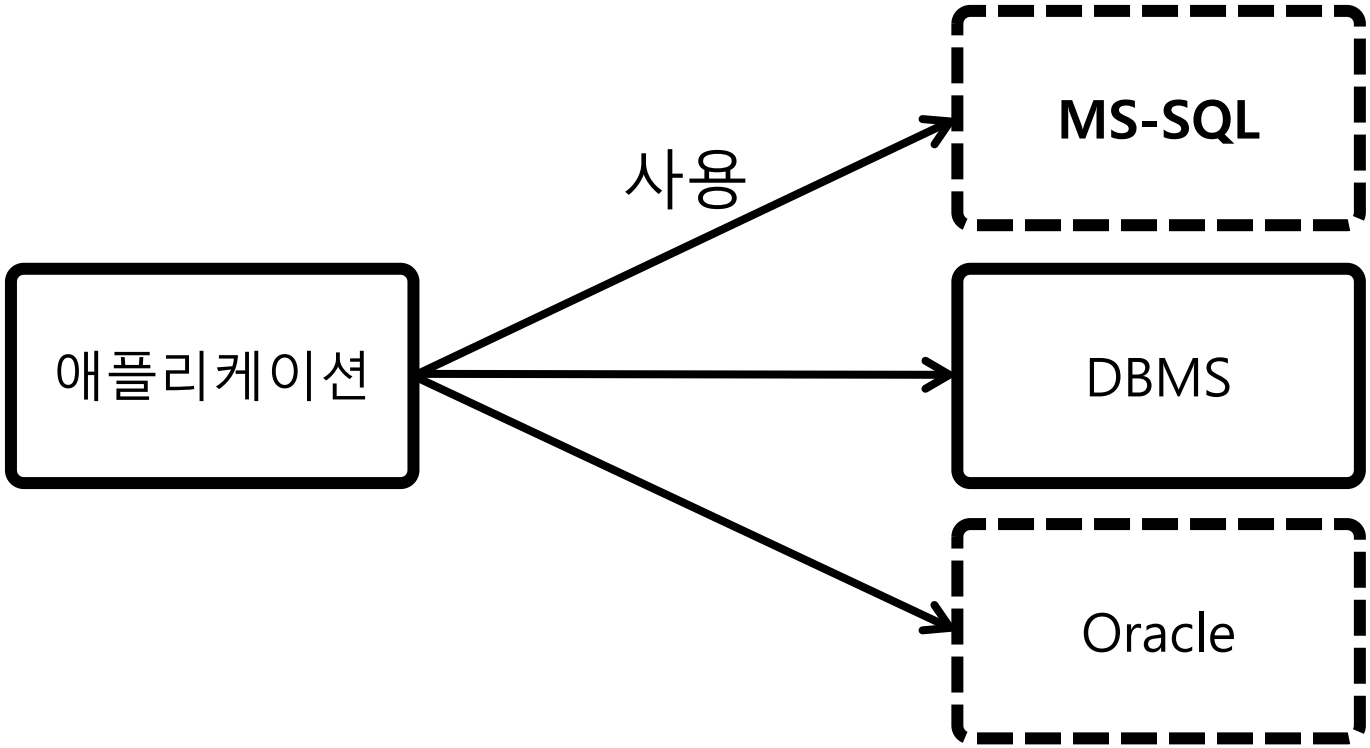
DBMS 사용 규칙

DBMS 사용 규칙

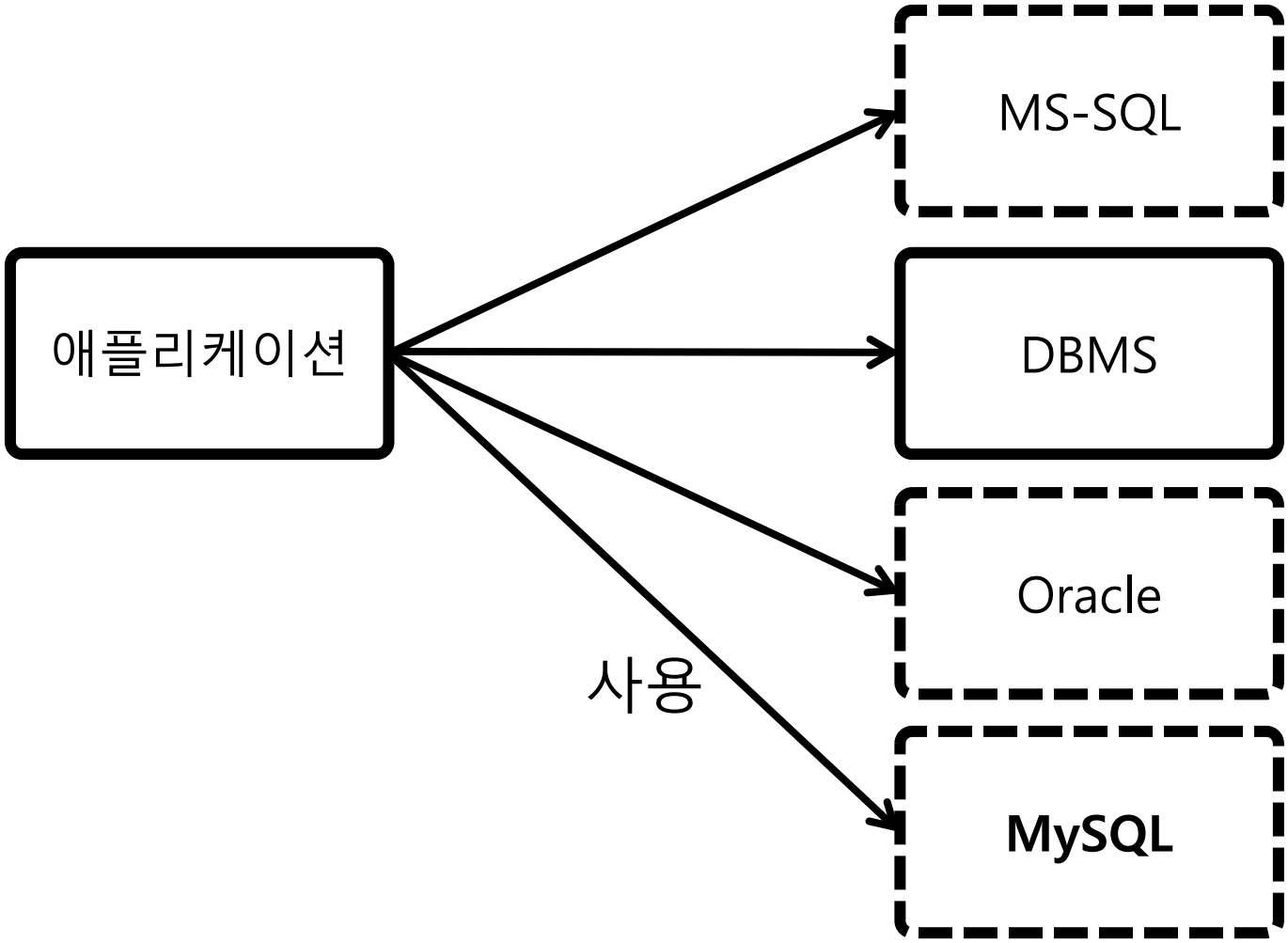
DBMS 사용 규칙



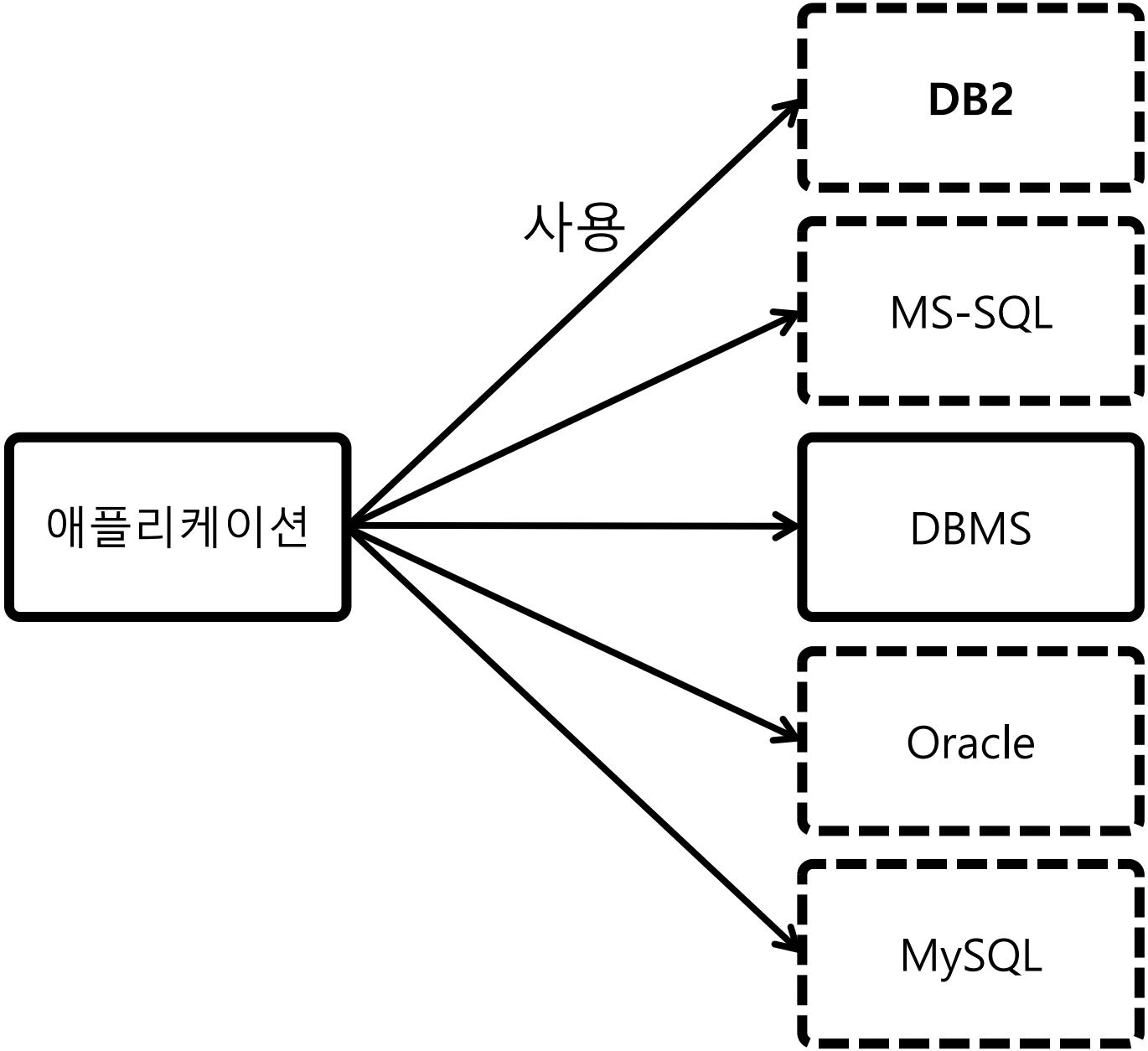
DBMS 사용 규칙



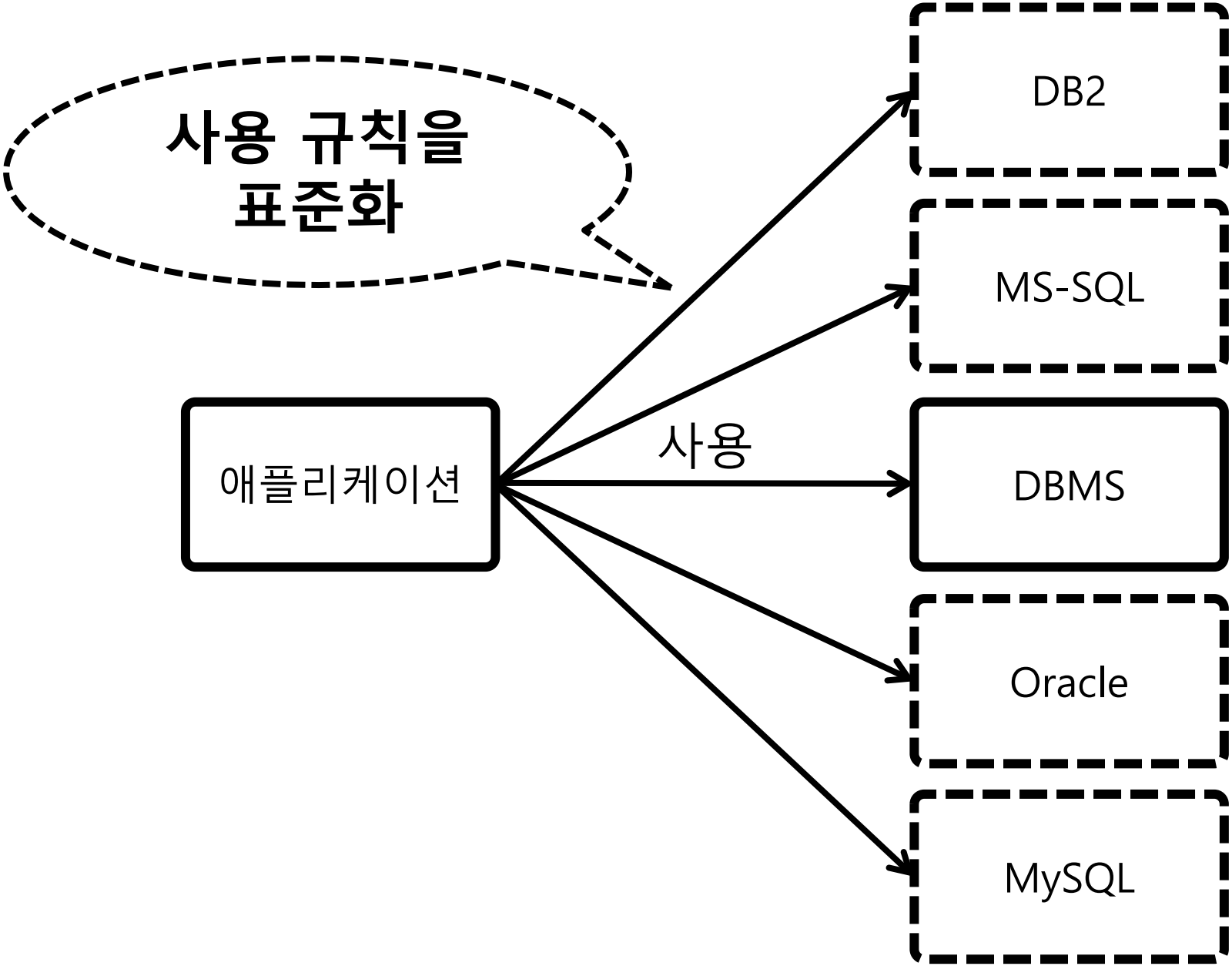
DBMS 사용 규칙



DBMS 사용 규칙

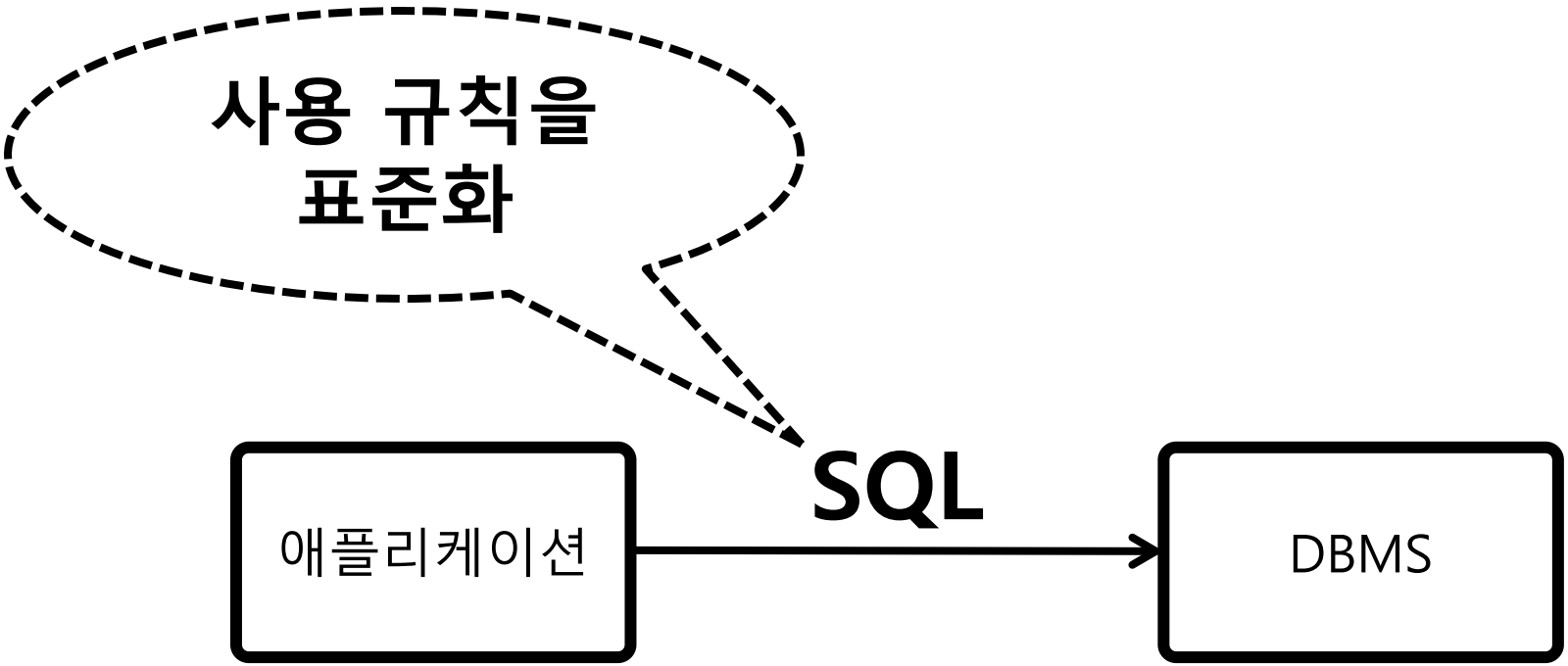


DBMS 사용 규칙

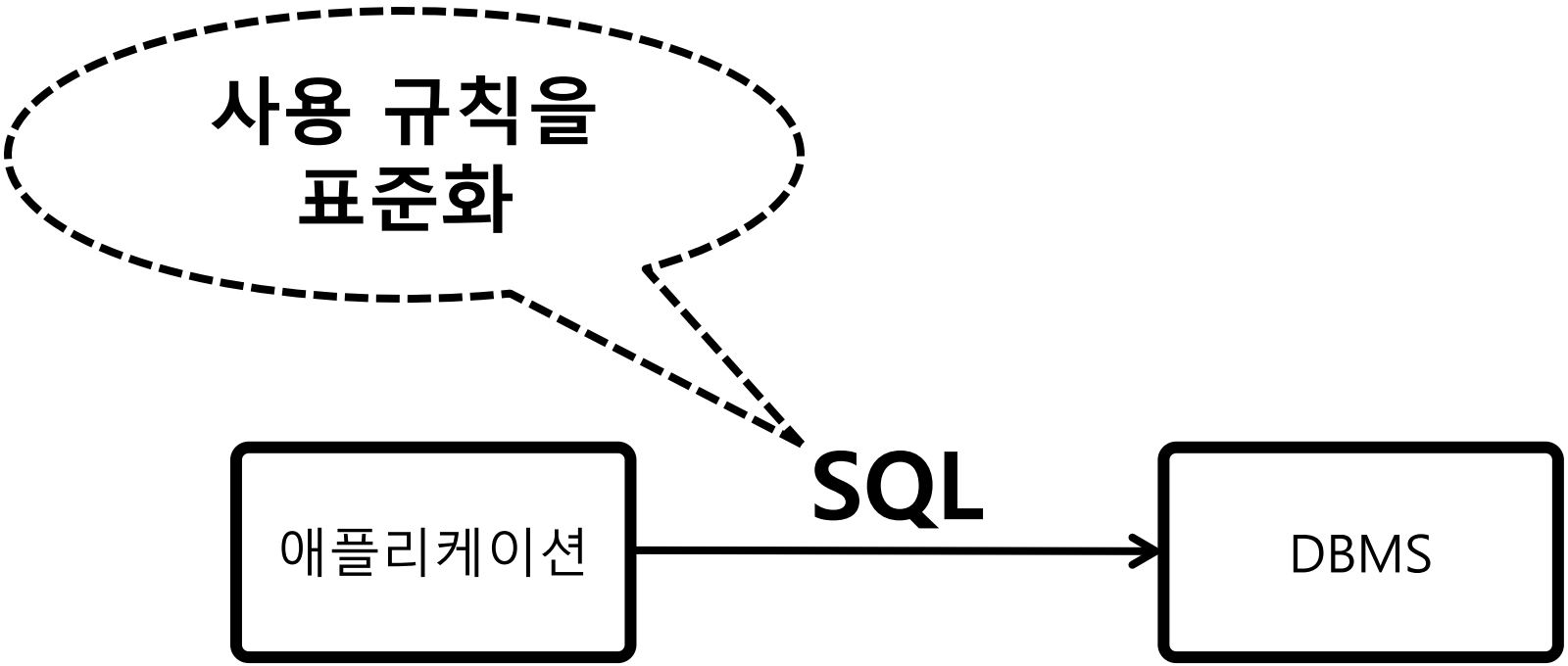


SQL

SQL



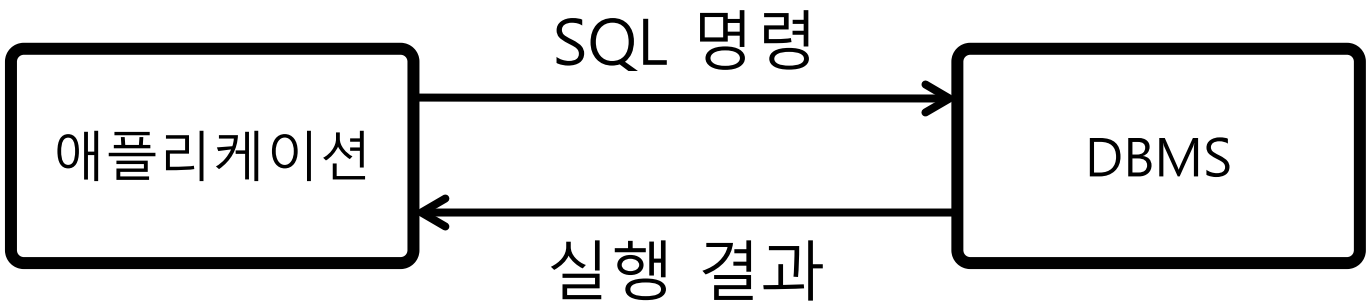
SQL



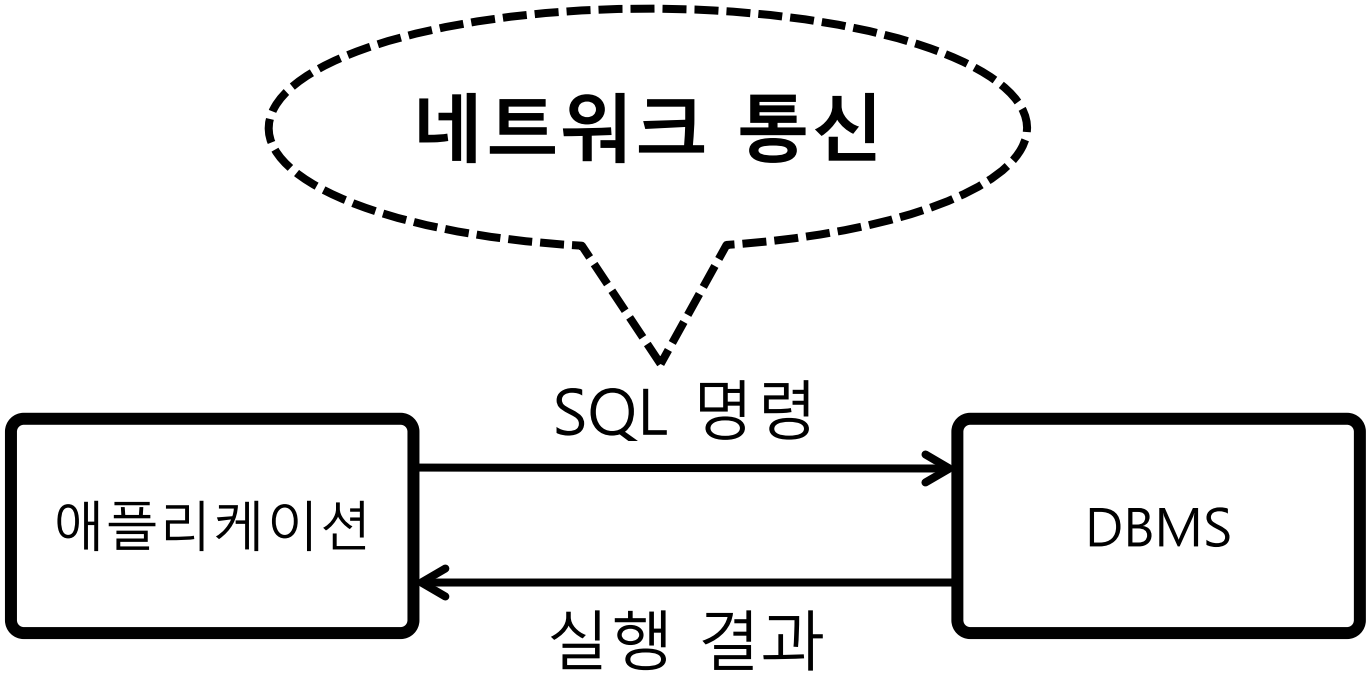
Structured Query Language

SQL 전달 및 데이터 수신

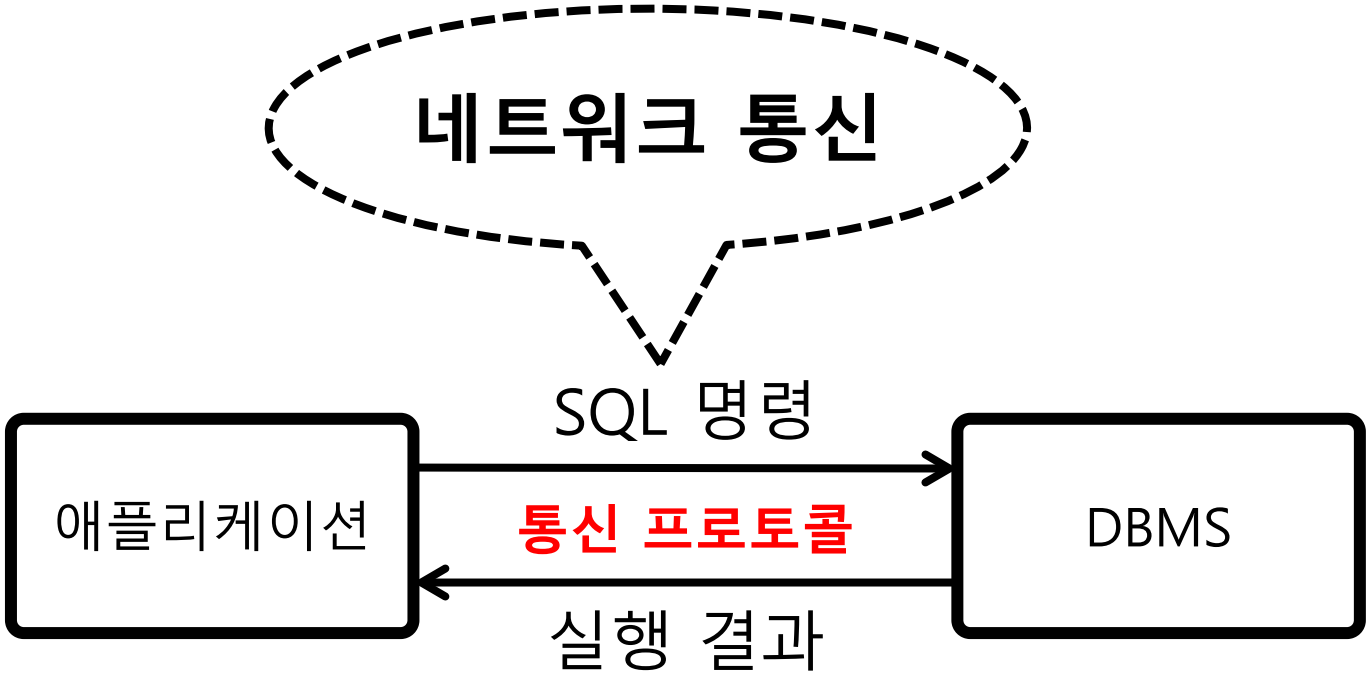
SQL 전송 프로토콜



SQL 전송 프로토콜

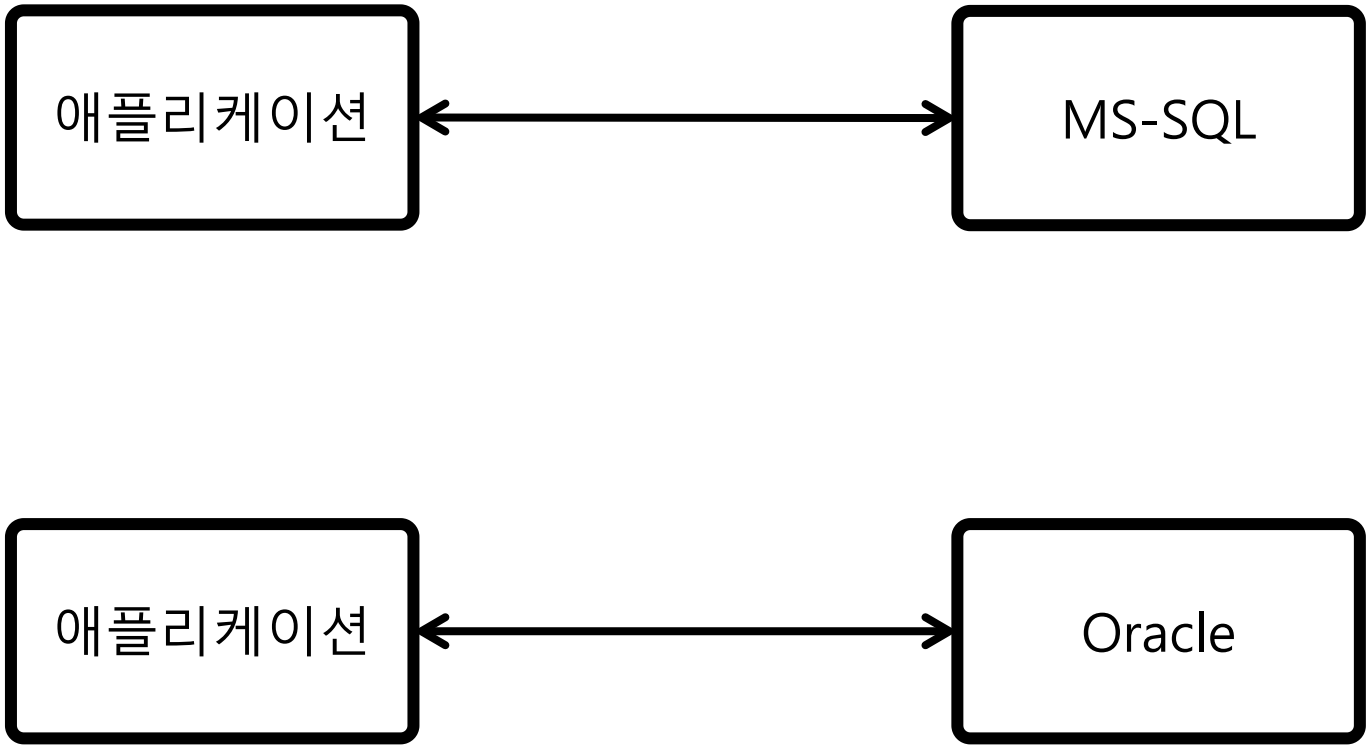


SQL 전송 프로토콜

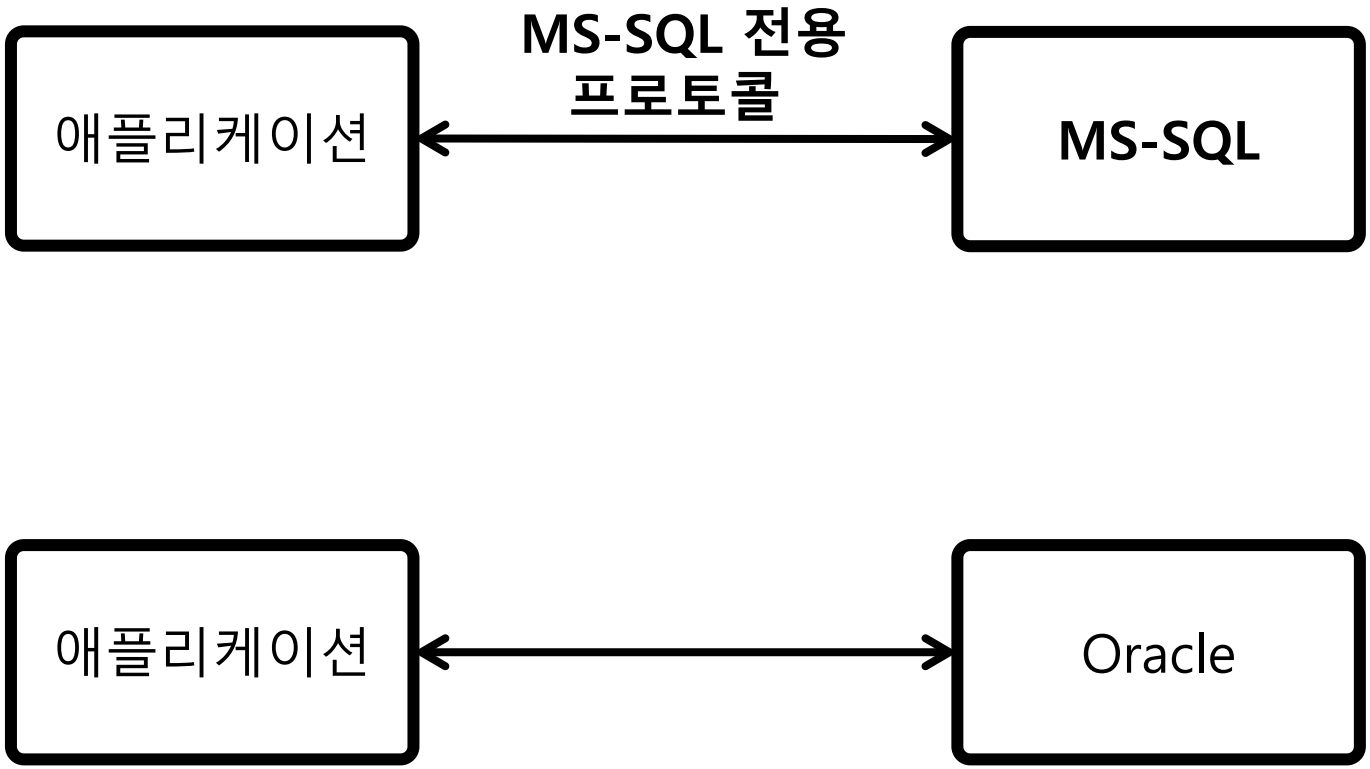


DBMS 전용 통신 프로토콜

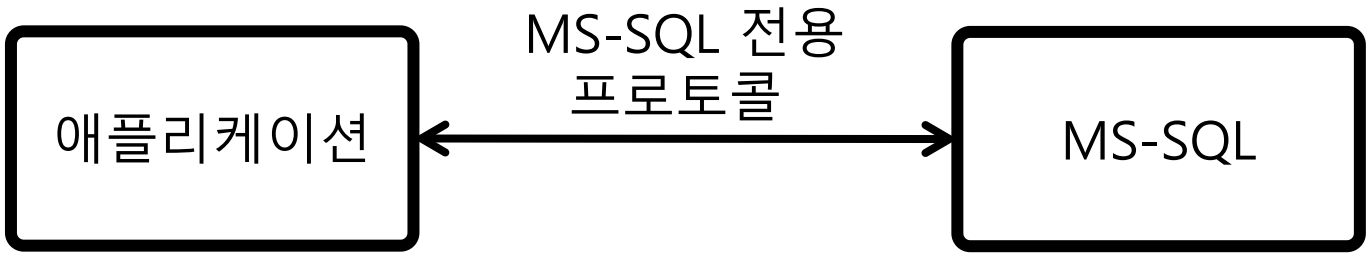
DBMS 전용 프로토콜



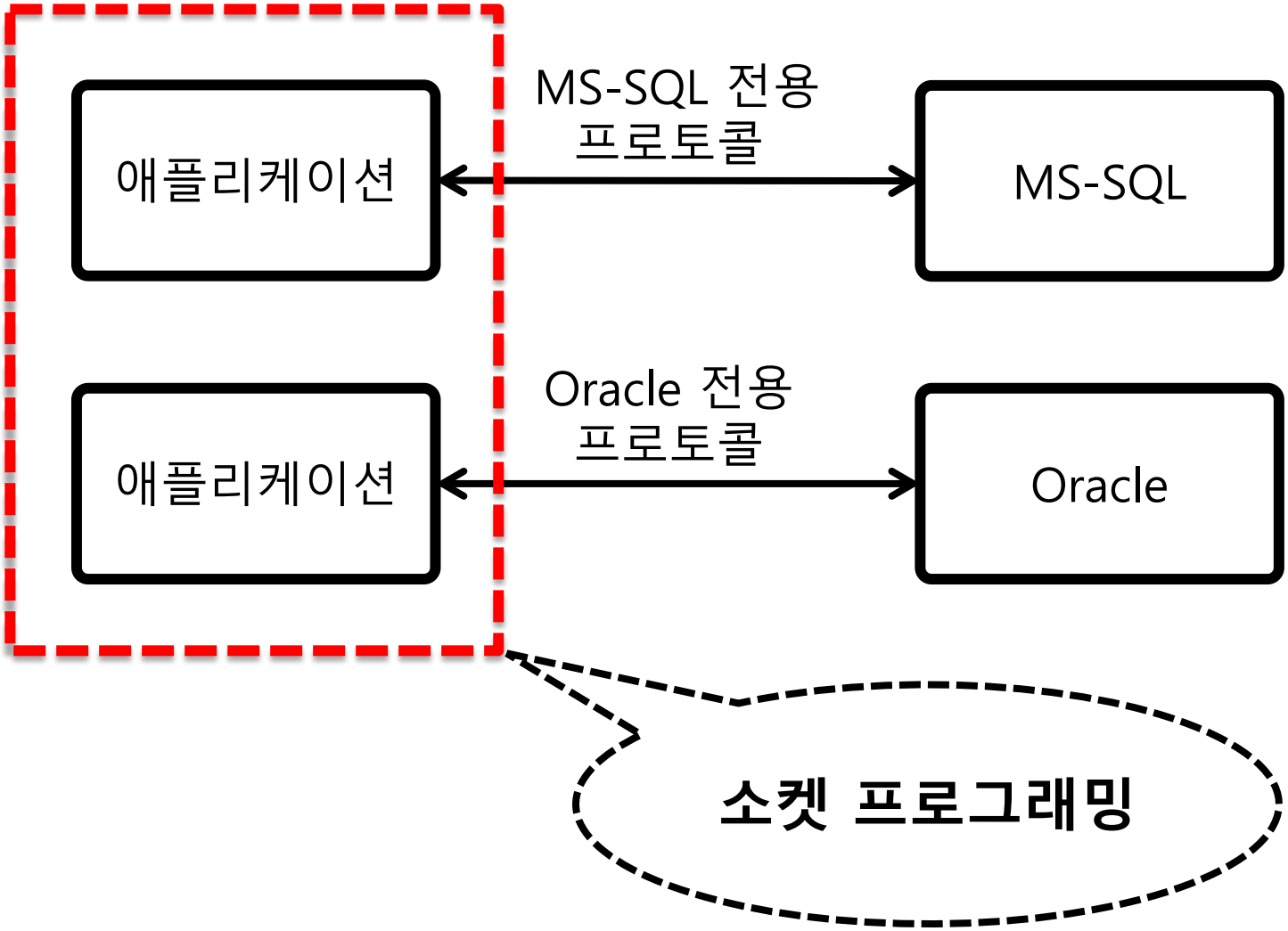
DBMS 전용 프로토콜



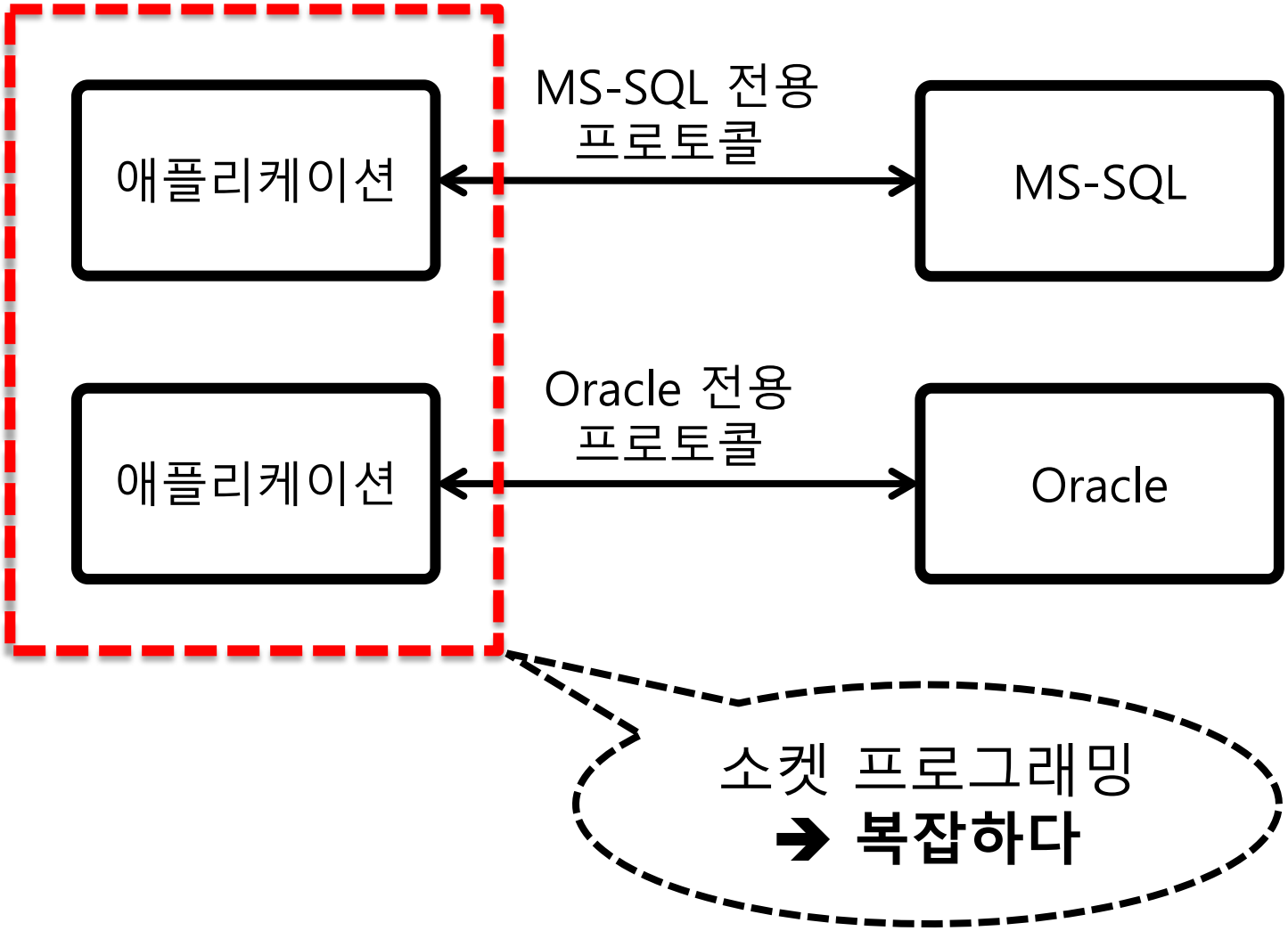
DBMS 전용 프로토콜



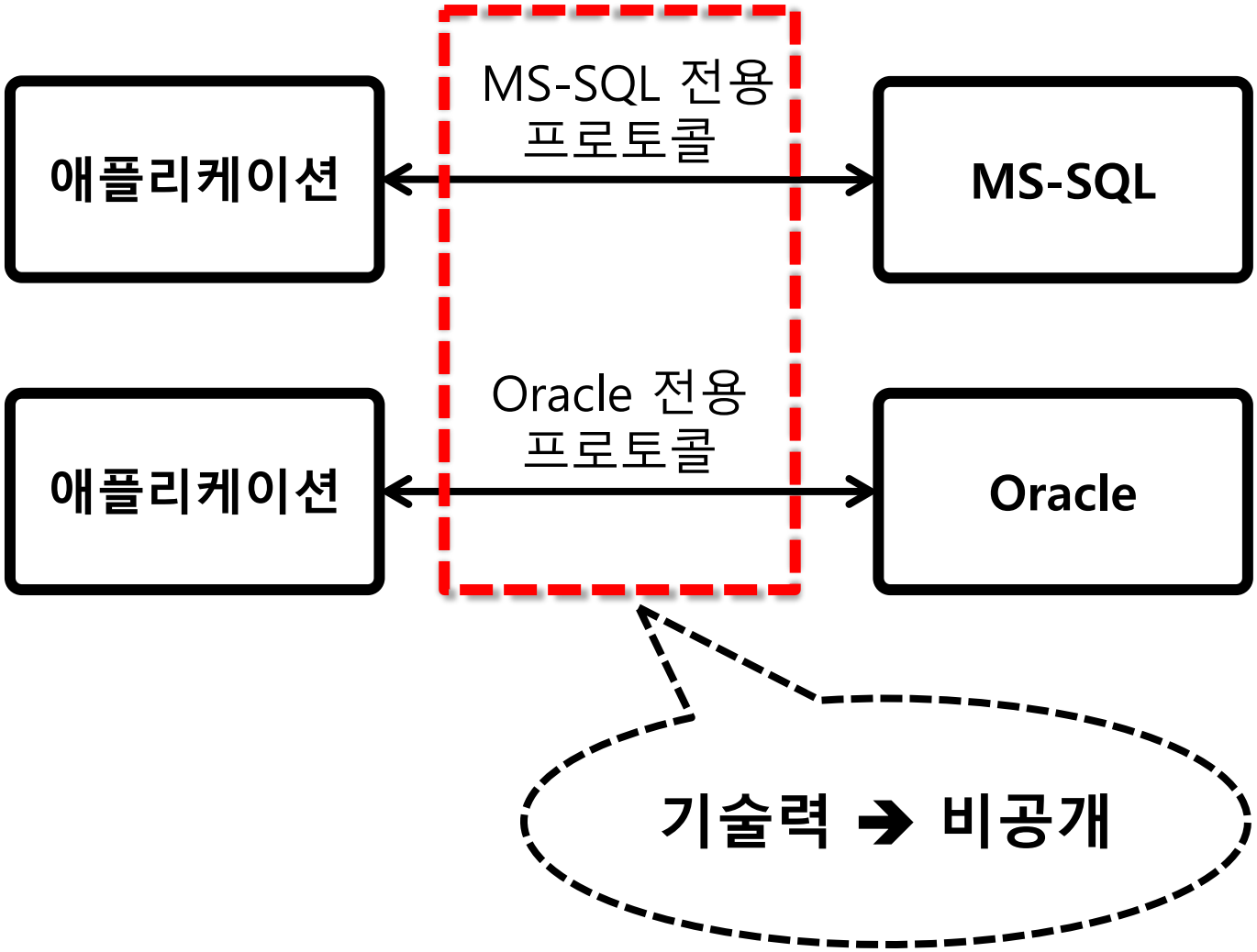
DBMS 전용 프로토콜


애플리케이션
MS-SQL 전용
프로토콜
MS-SQL
애플리케이션
Oracle 전용
프로토콜
Oracle
소켓 프로그래밍

DBMS 전용 프로토콜

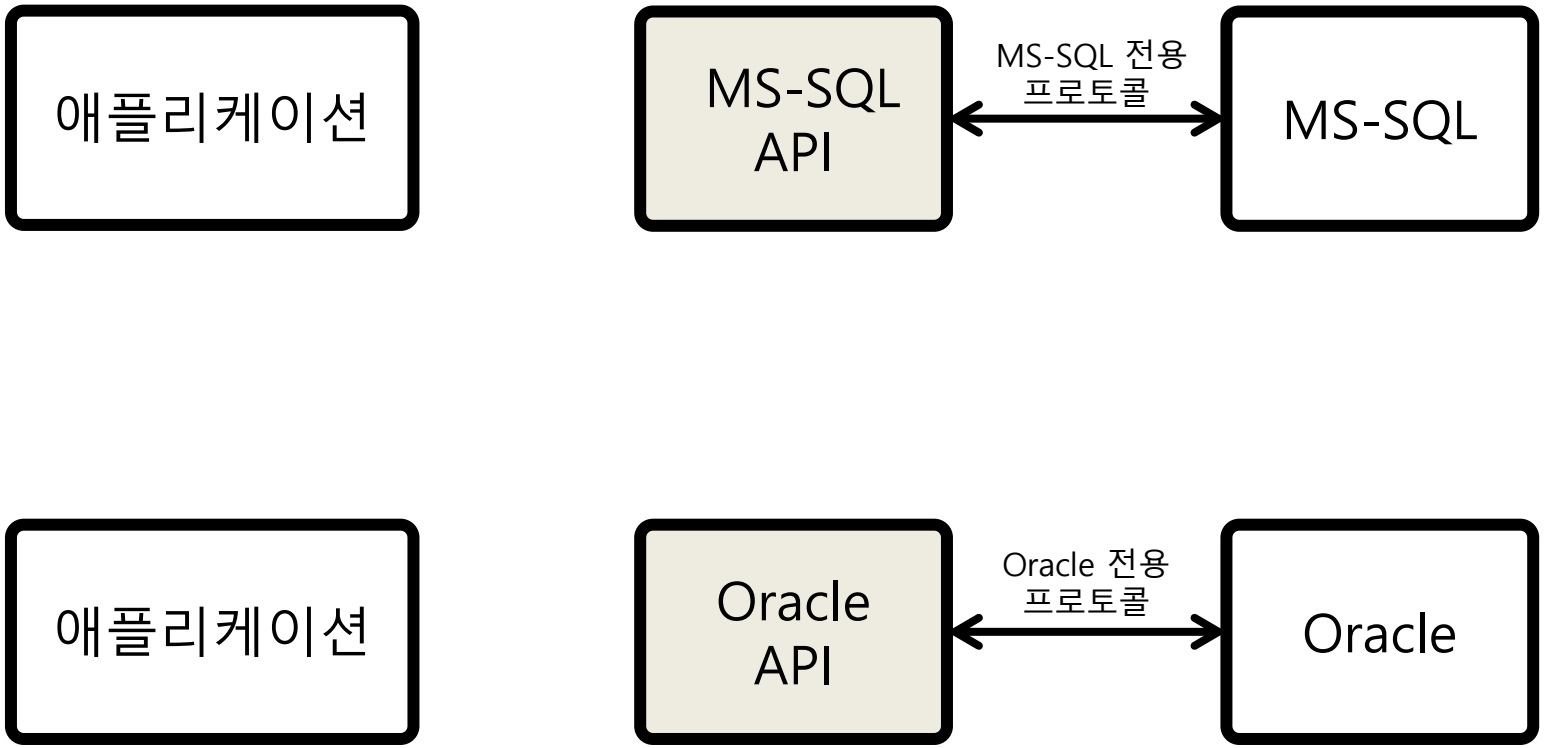

애플리케이션
MS-SQL 전용
프로토콜
MS-SQL
애플리케이션
Oracle 전용
프로토콜
Oracle
소켓 프로그래밍
➔ 복잡하다

DBMS 전용 프로토콜

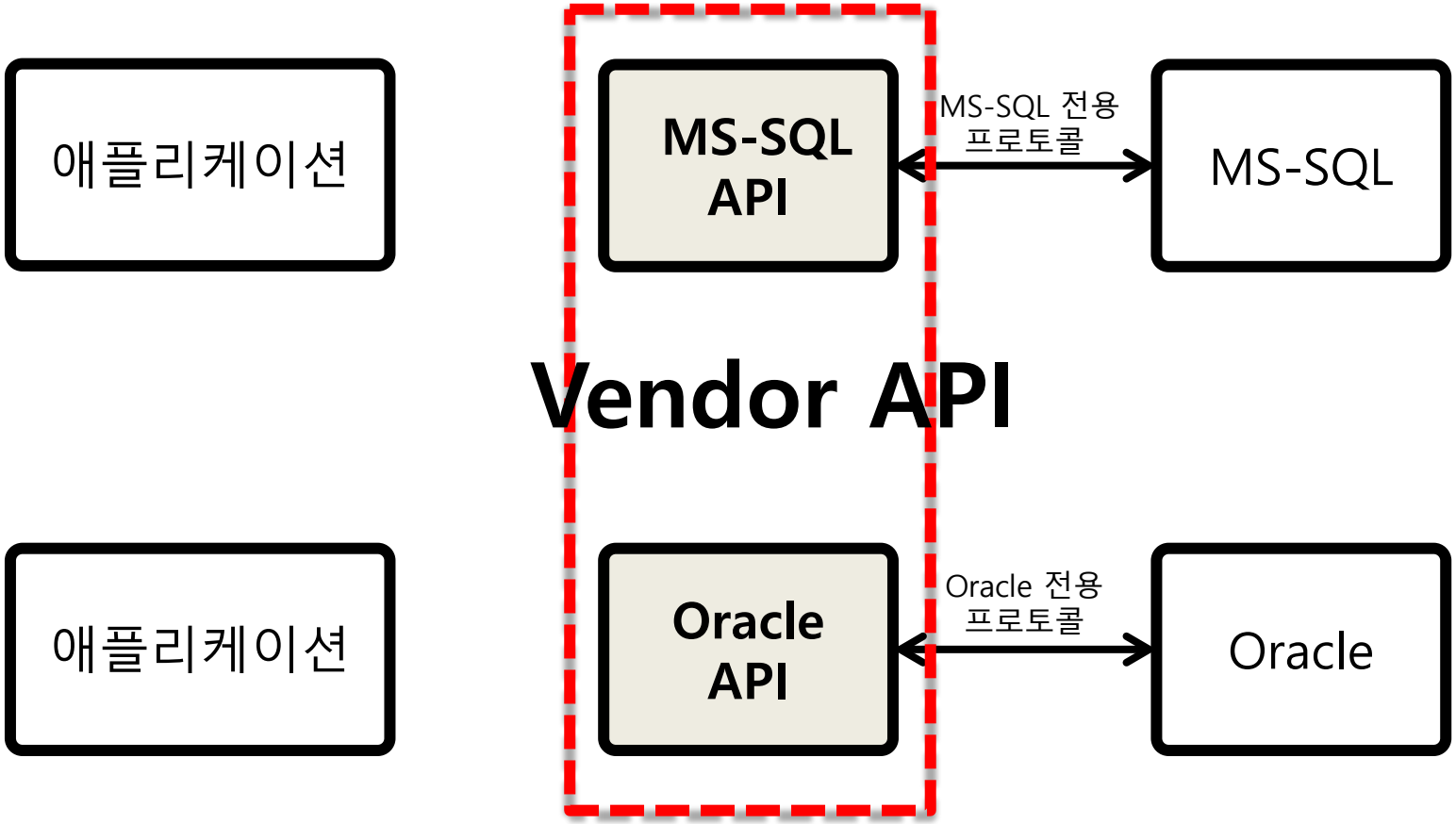

MS-SQL 전용
프로토콜
애플리케이션
MS-SQL
Oracle 전용
프로토콜
애플리케이션
Oracle
기술력 ➔ 비공개

Vendor(Native) API

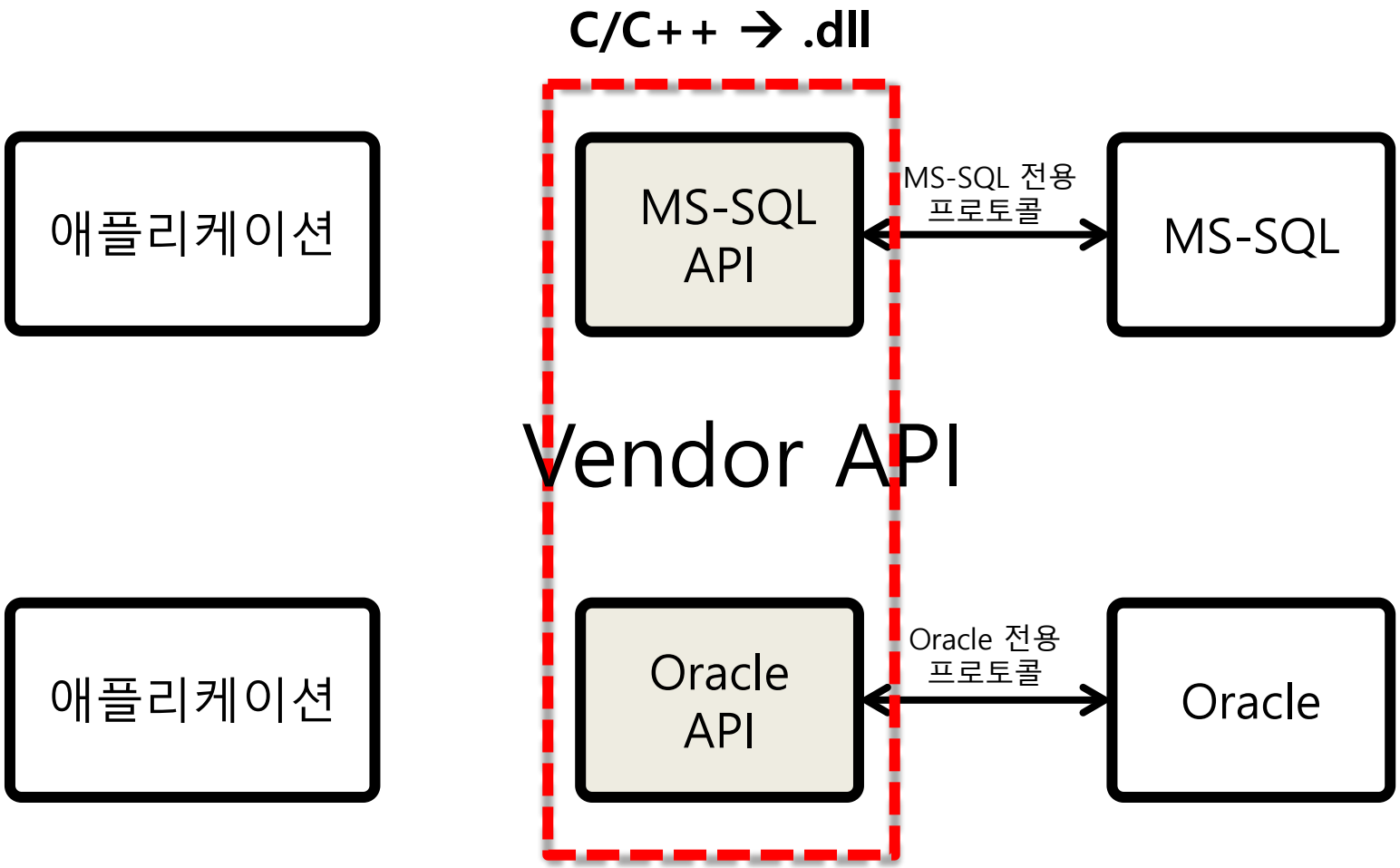
Vendor(Native) API



Vendor(Native) API

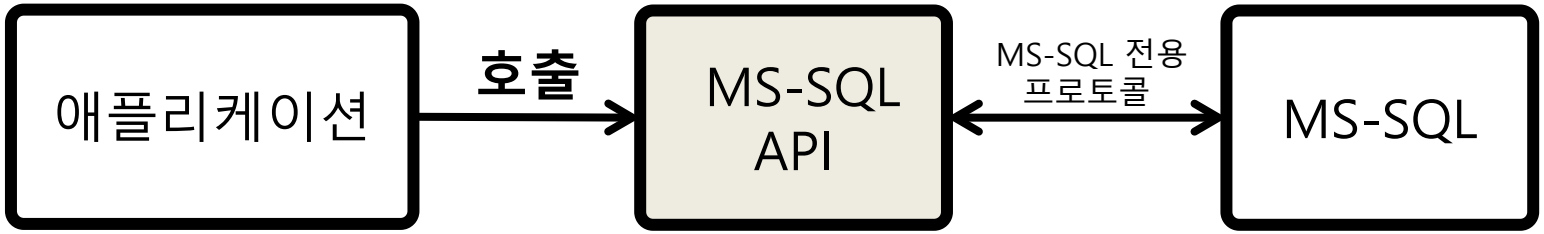


Vendor(Native) API

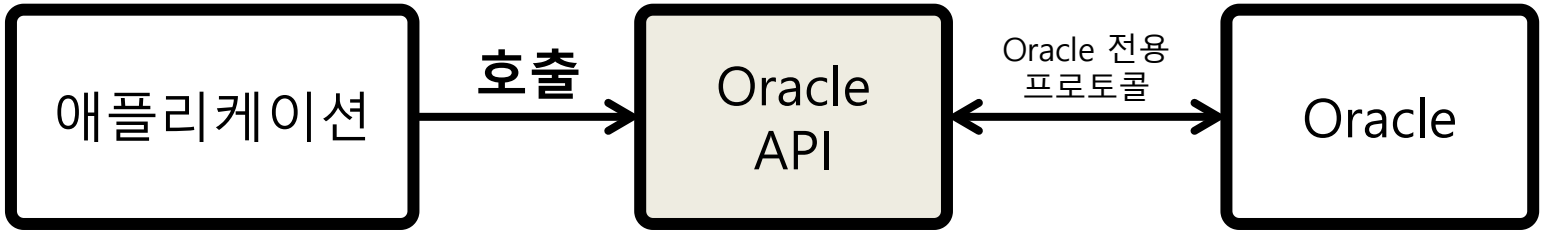


Vendor(Native) API

C/C++ → .dll



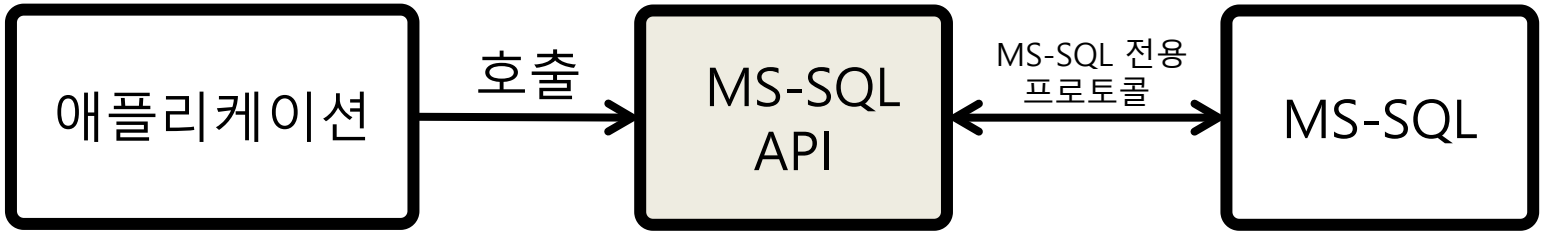
Vendor API



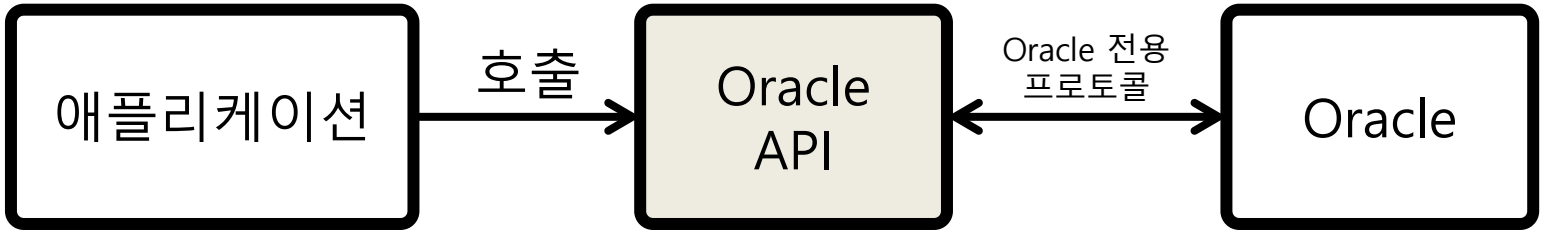
DBMS에 종속

DBMS에 종속

C/C++ → .dll

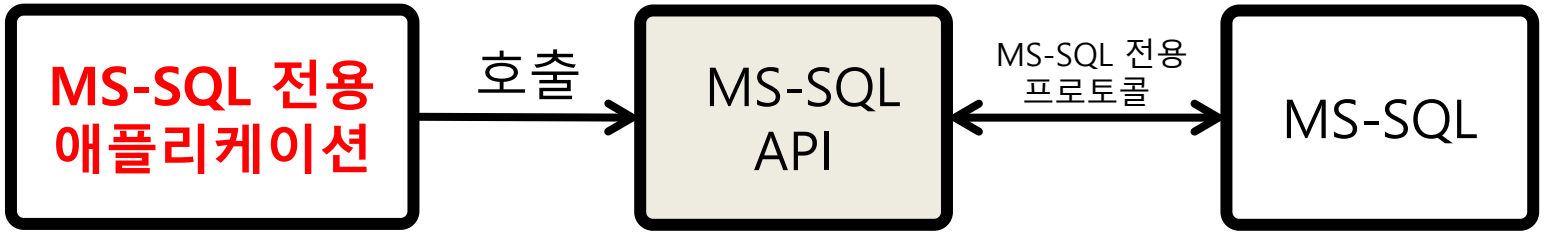


Vendor API

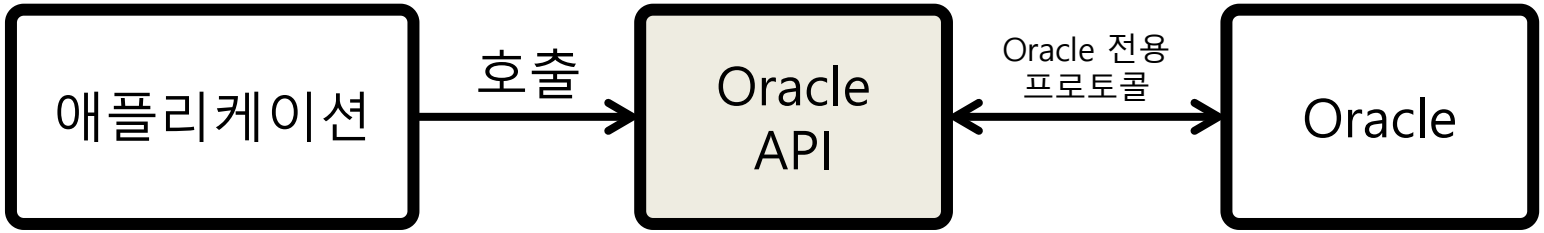


DBMS에 종속

C/C++ → .dll

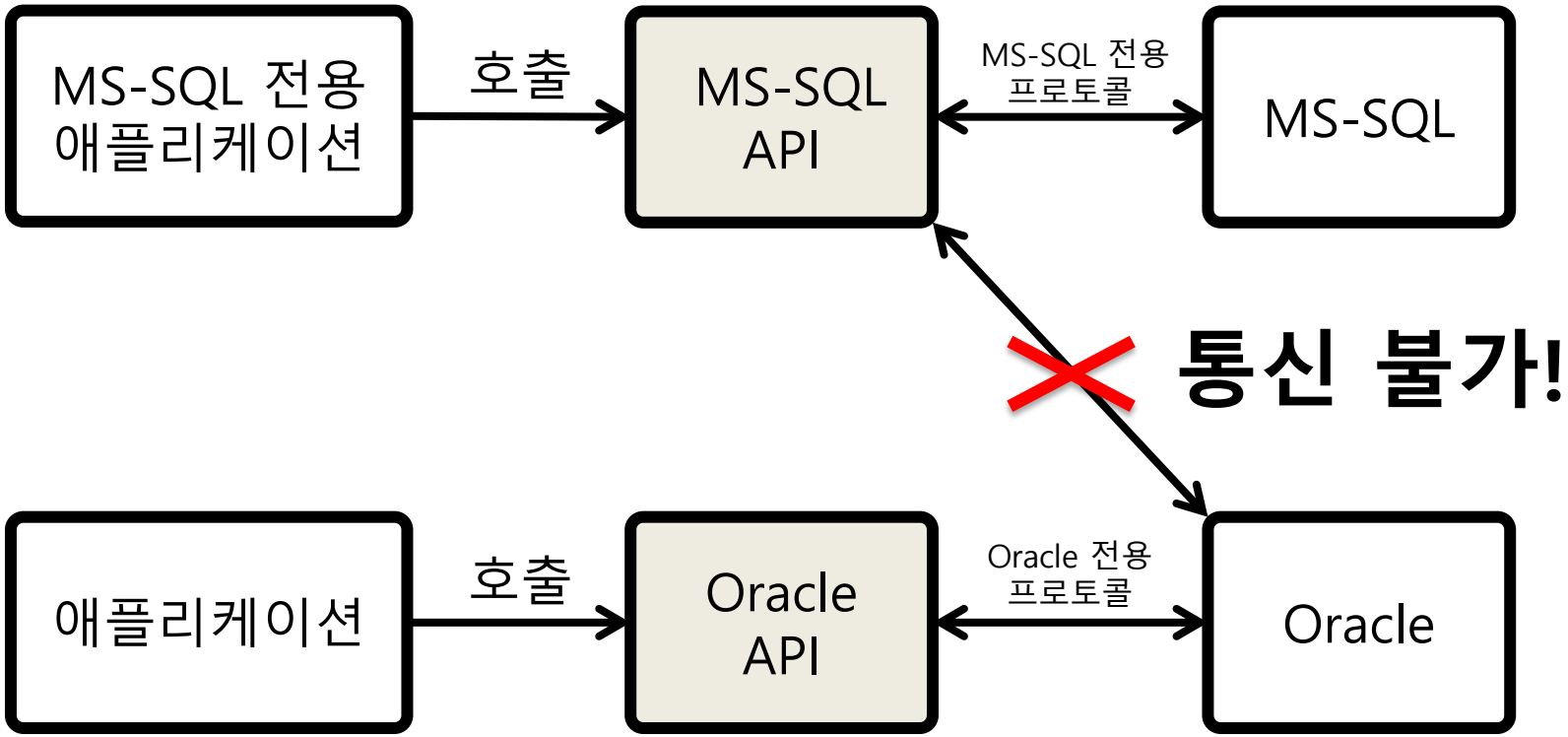


Vendor API



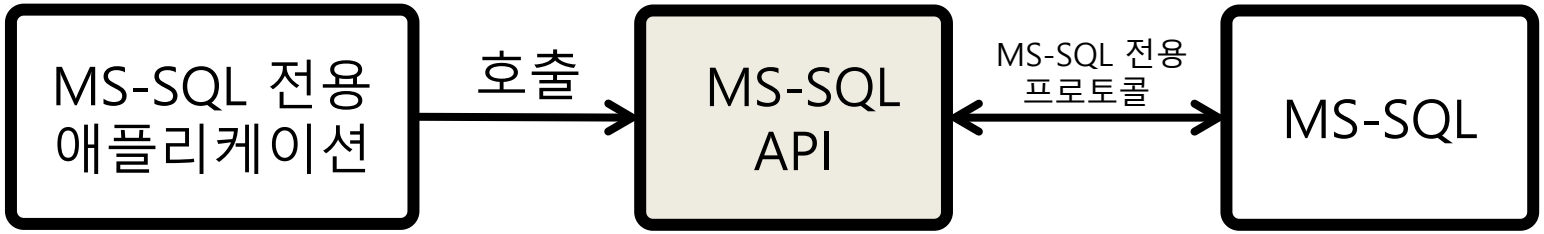
DBMS에 종속

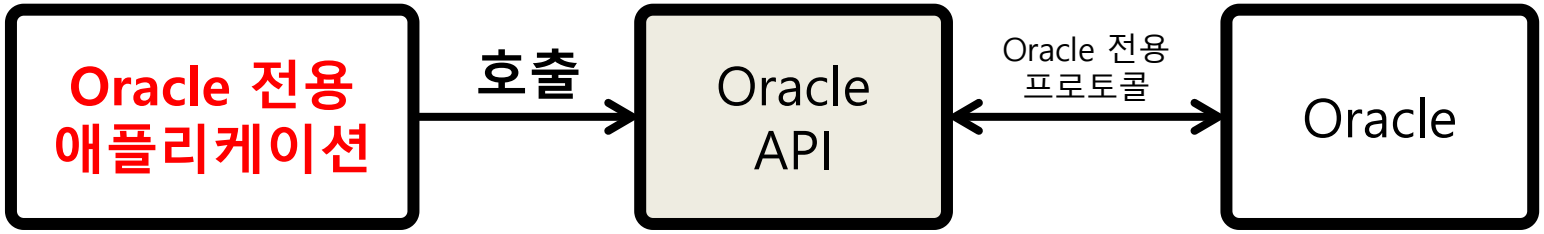
C/C++ → .dll



DBMS에 종속

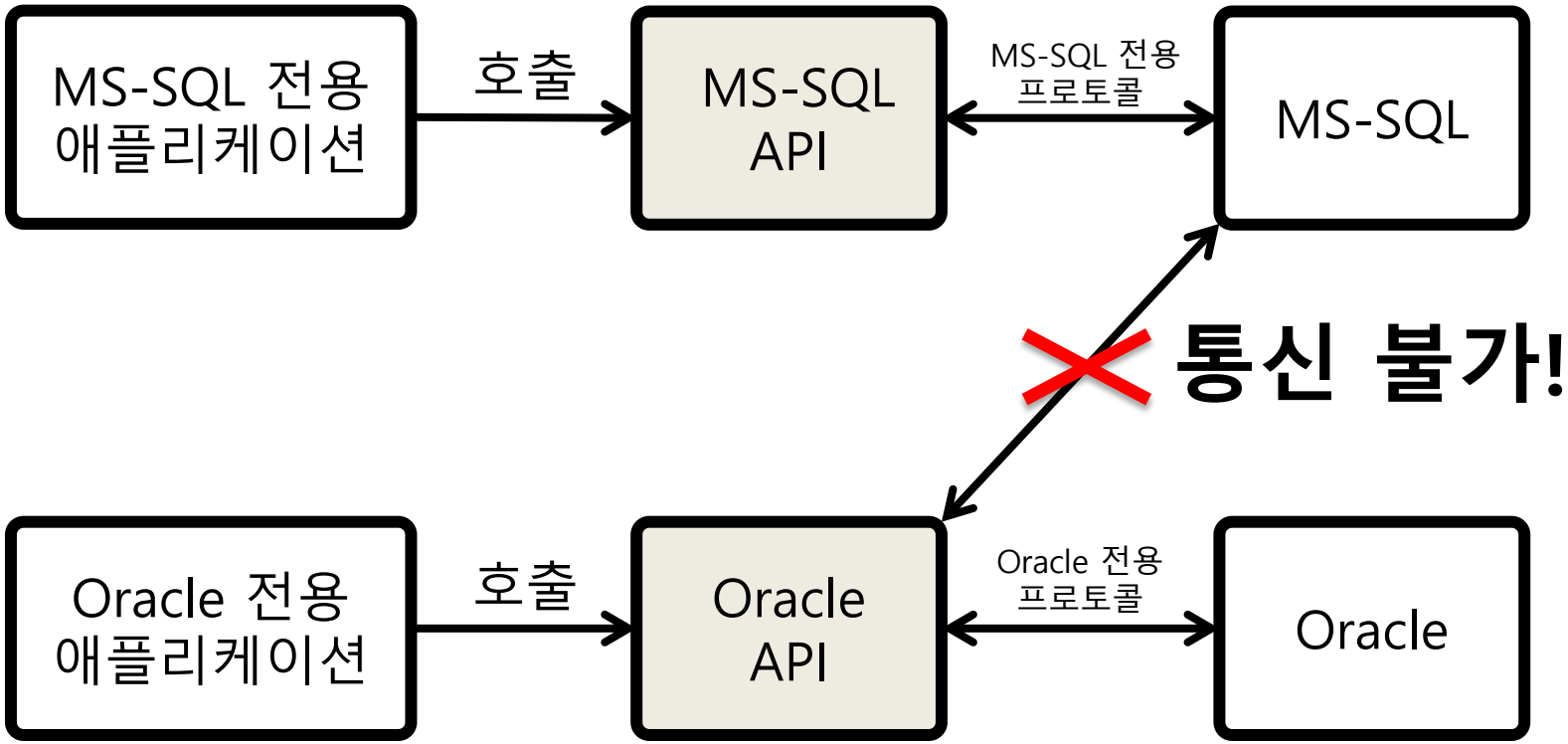
C/C++ → .dll



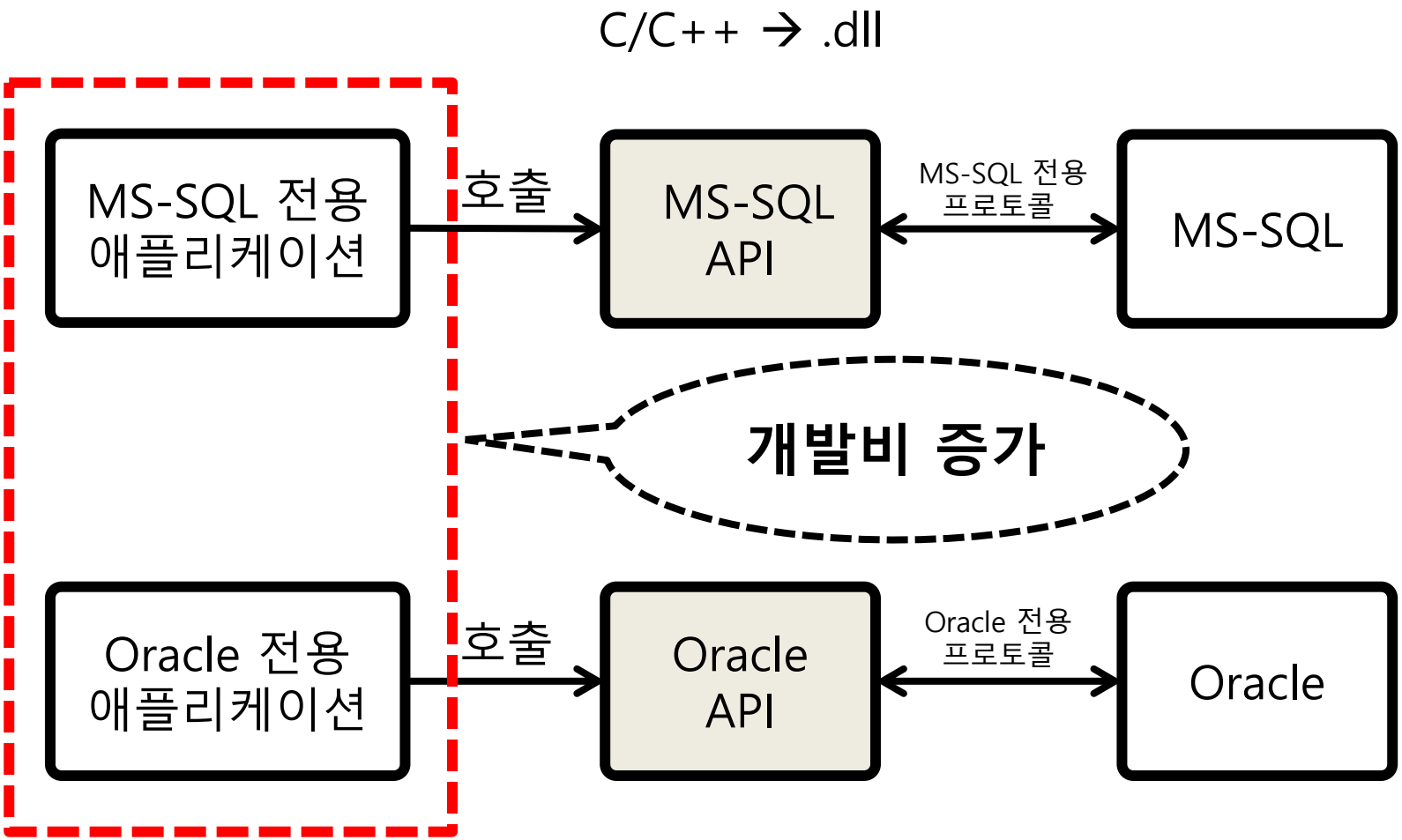


DBMS에 종속

C/C++ → .dll



DBMS에 종속


C/C++ → .dll
MS-SQL 전용
애플리케이션
호출
MS-SQL
API
MS-SQL 전용
프로토콜
MS-SQL
개발비 증가
Oracle 전용
애플리케이션
호출
Oracle
API
Oracle 전용
프로토콜
Oracle

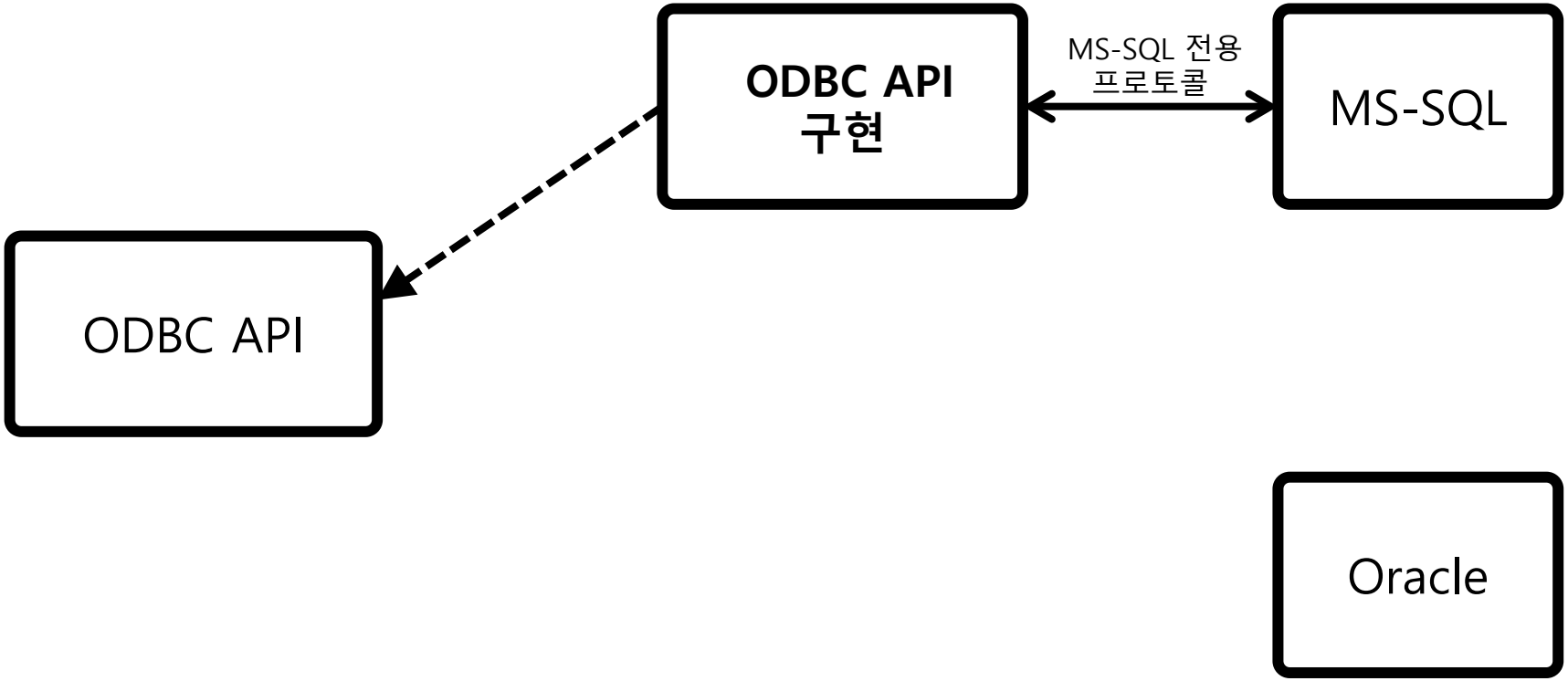
ODBC 등장

Open DataBase Connectivity

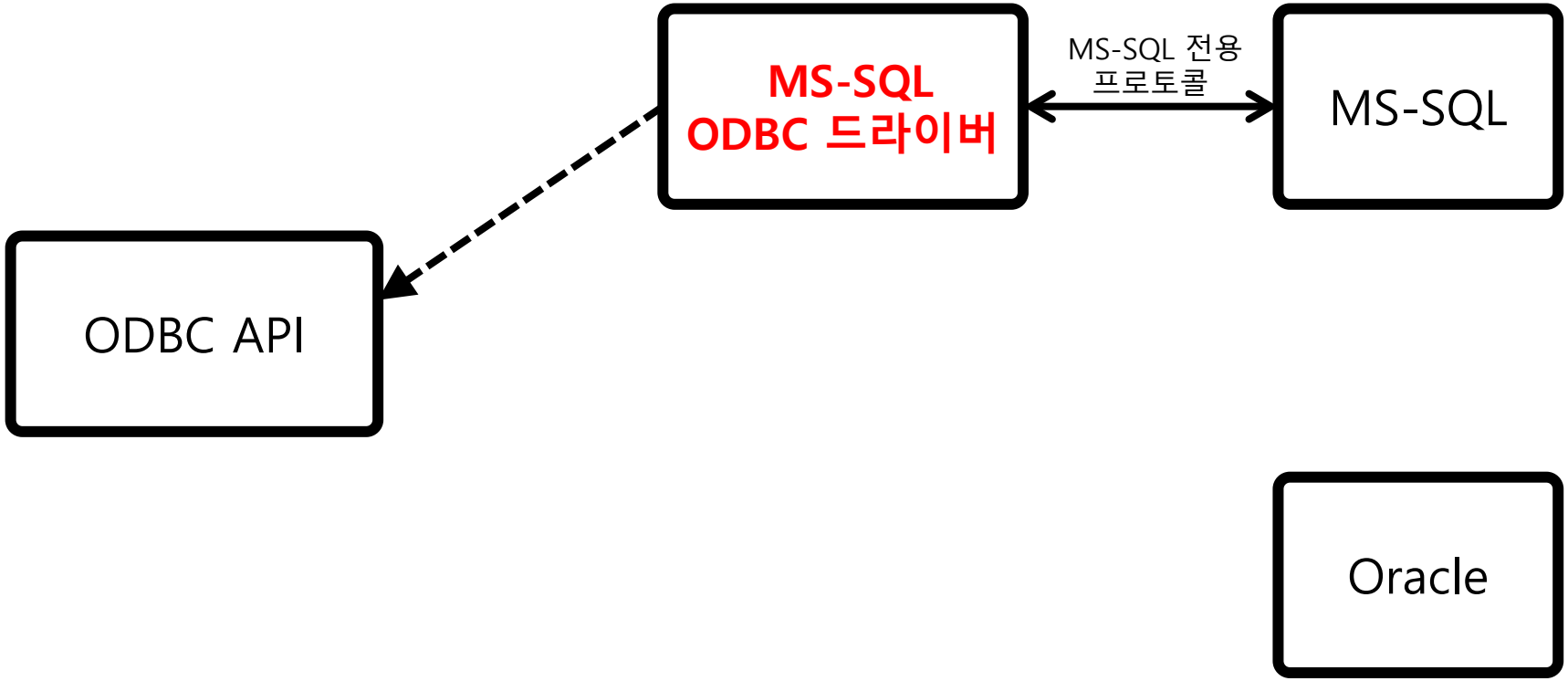
DBMS에 접근하기 위한 표준 인터페이스

ODBC

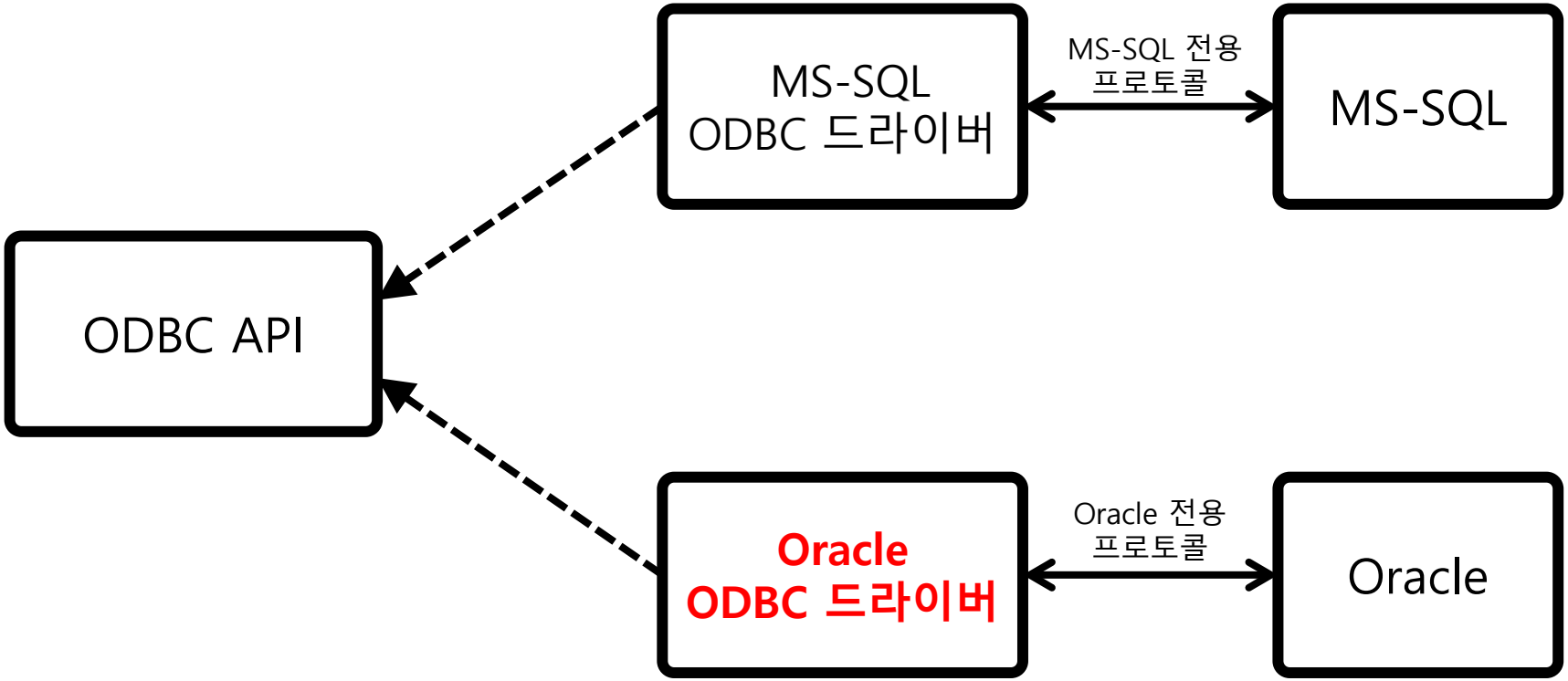
ODBC



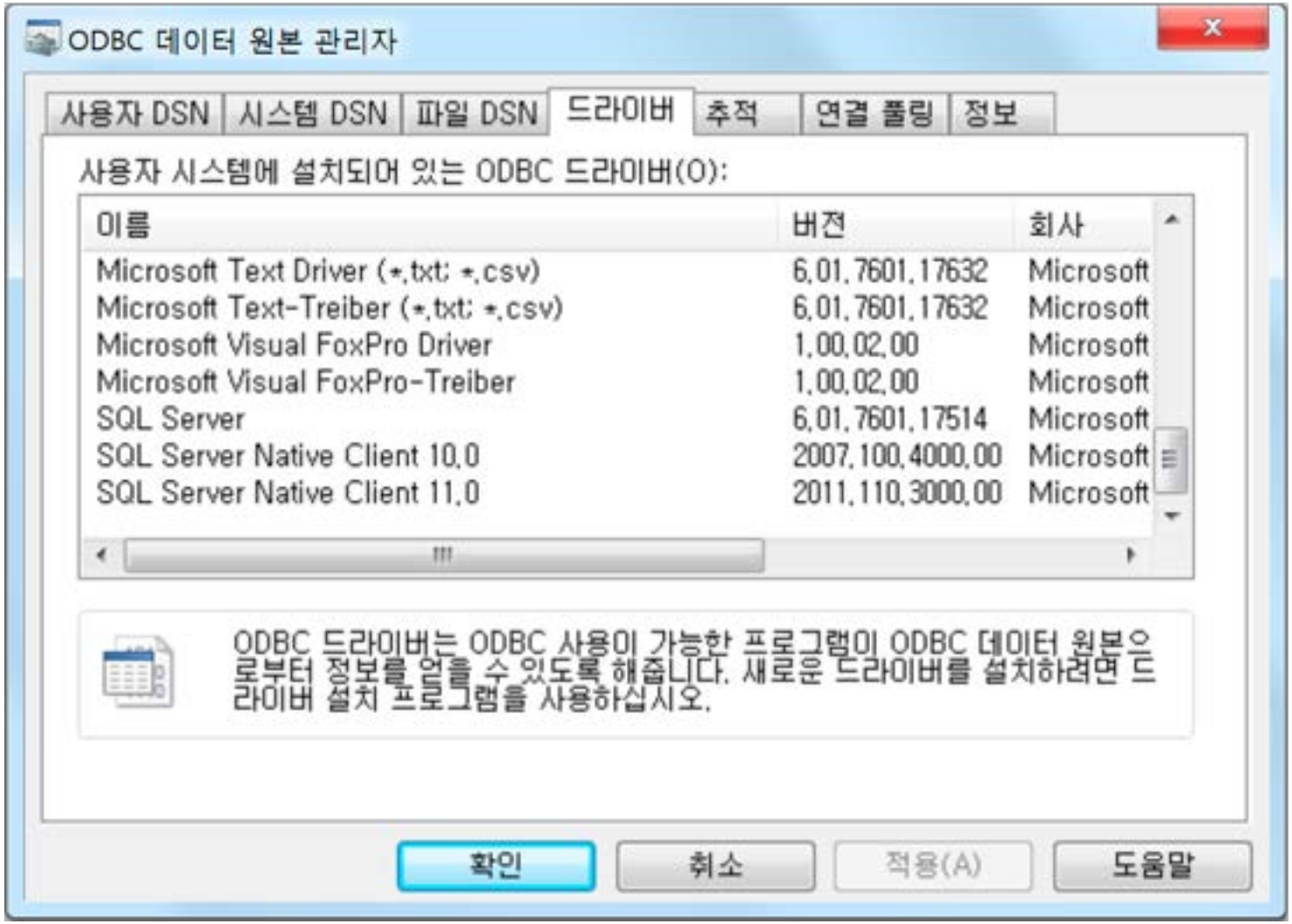
ODBC



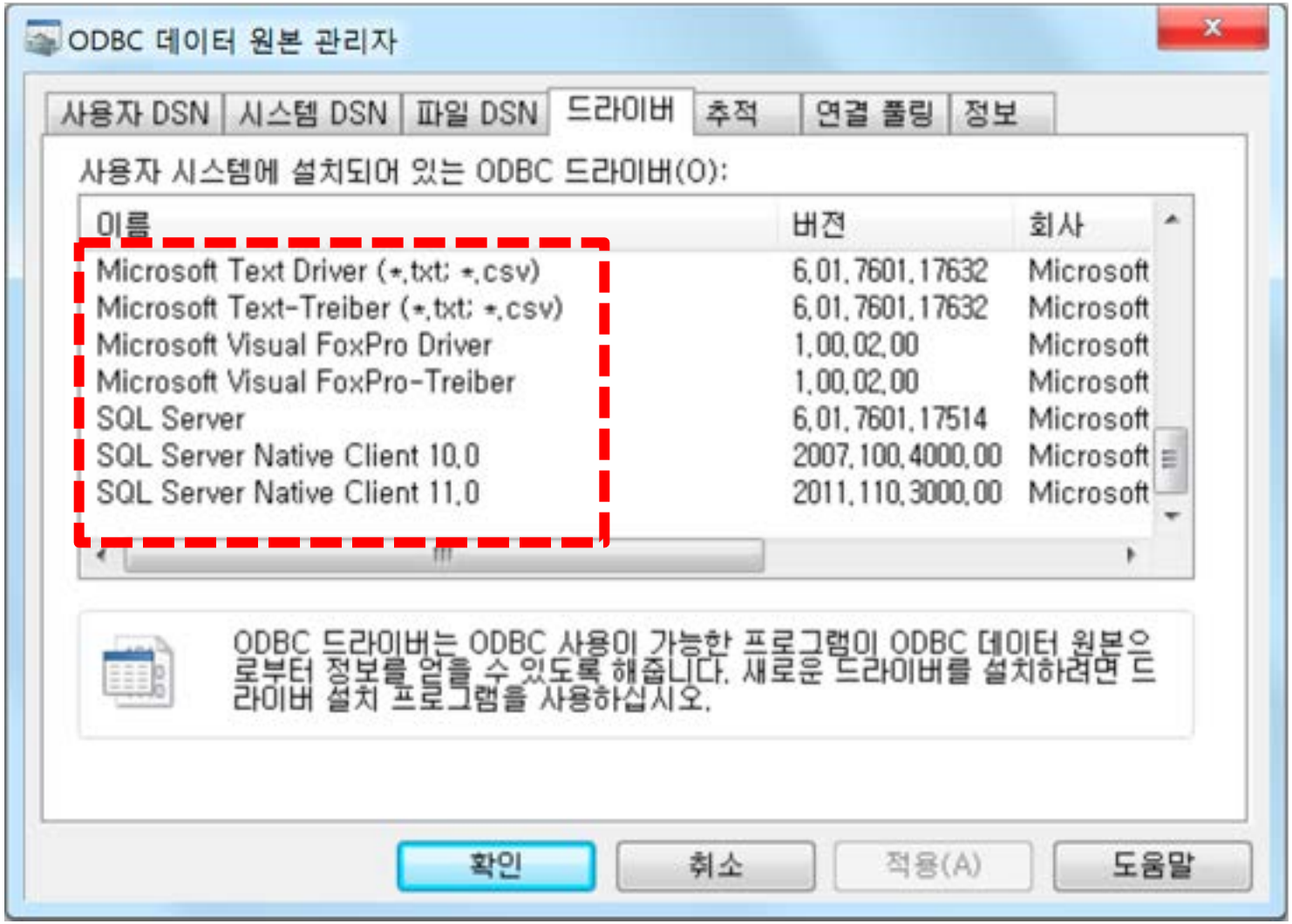
ODBC


MS-SQL
ODBC 드라이버
MS-SQL 전용
프로토콜
MS-SQL
ODBC API
Oracle
ODBC 드라이버
Oracle 전용
프로토콜
Oracle

ODBC 드라이버 관리창

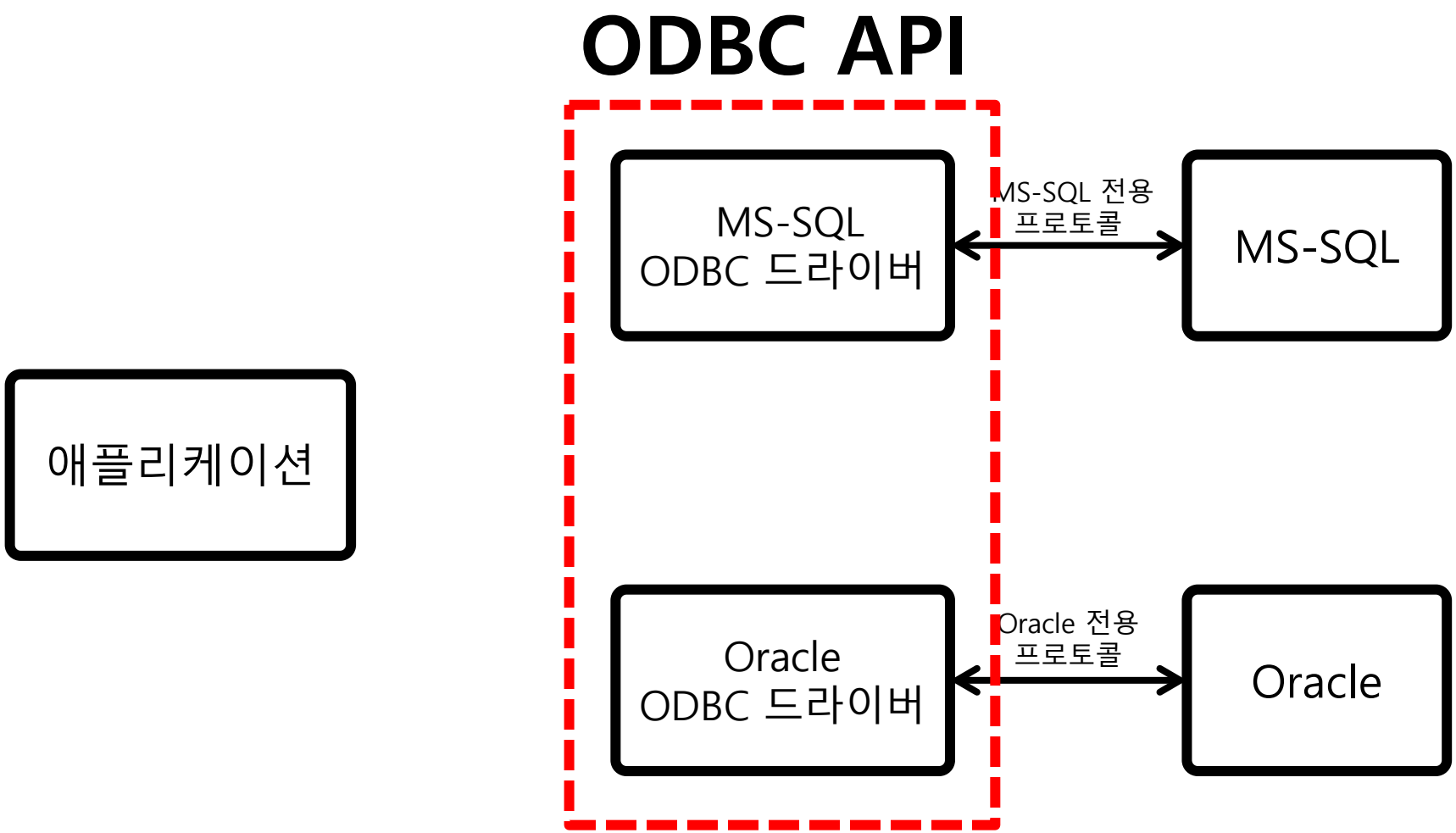


ODBC 드라이버 관리창

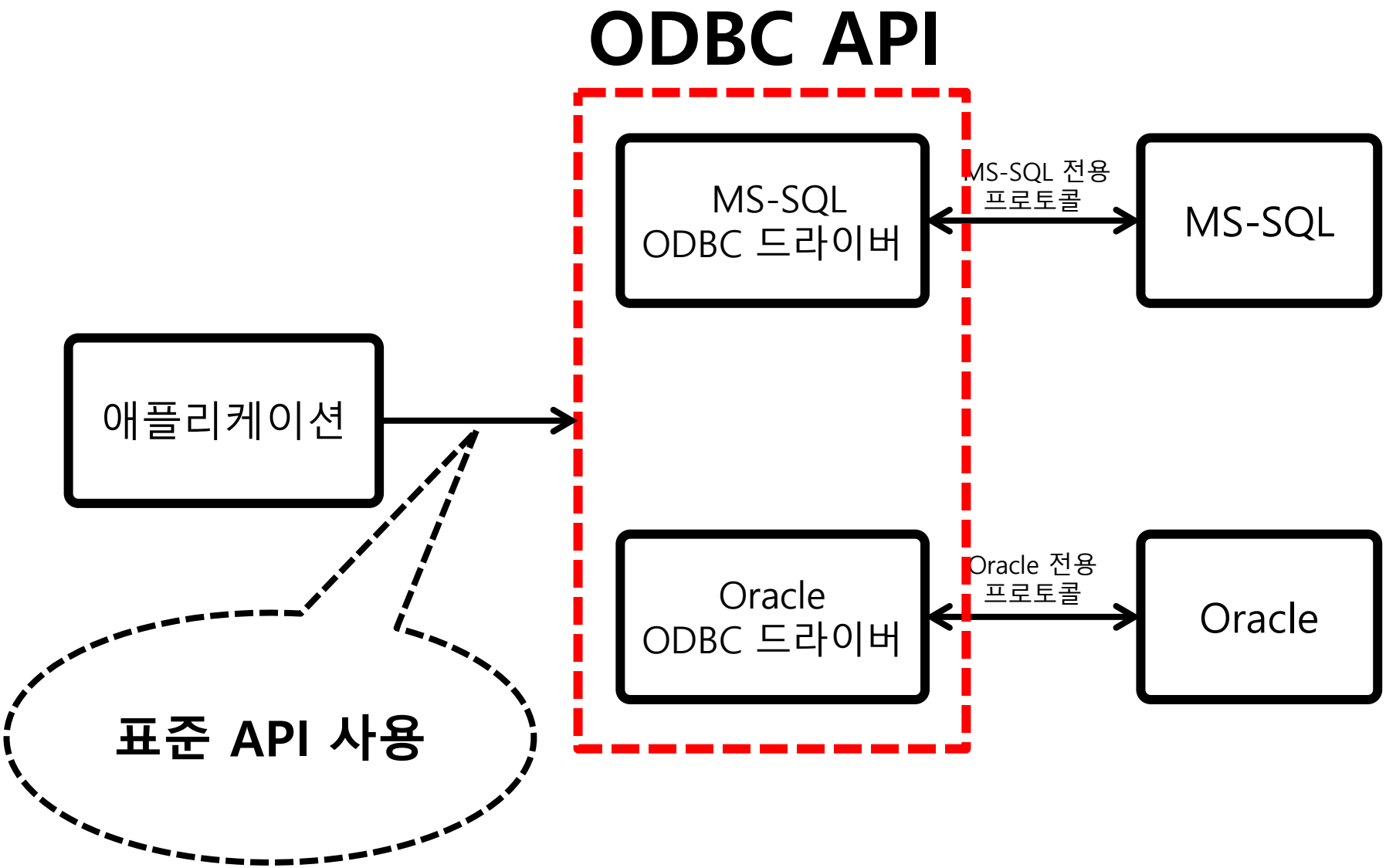


DBMS 종속 탈피

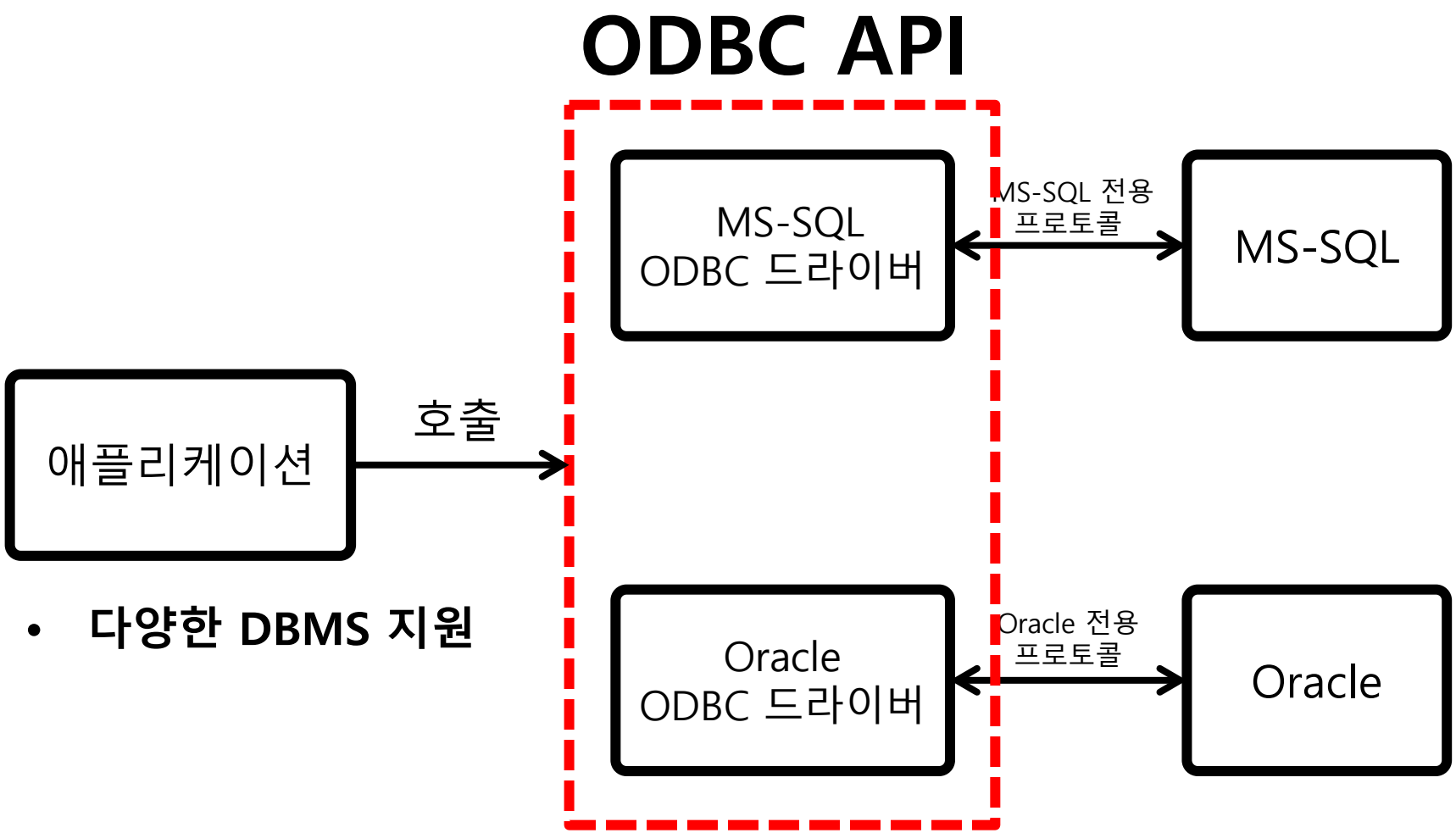
DBMS 종속 탈피


ODBC API
MS-SQL
ODBC 드라이버
MS-SQL 전용
프로토콜
MS-SQL
애플리케이션
Oracle
ODBC 드라이버
Oracle 전용
프로토콜
Oracle

DBMS 종속 탈피

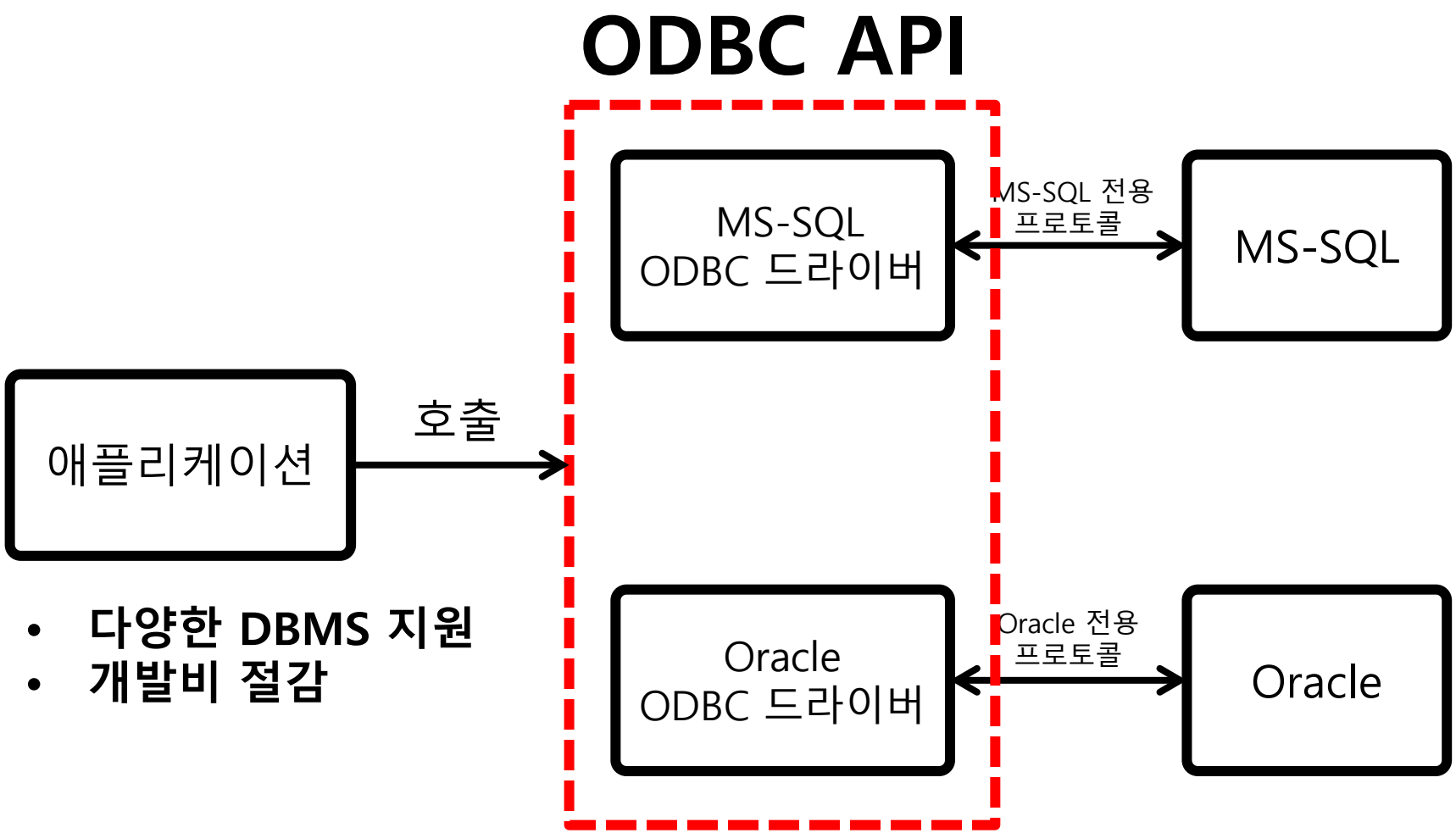

ODBC API
MS-SQL
ODBC 드라이버
MS-SQL 전용
프로토콜
MS-SQL
애플리케이션
Oracle
ODBC 드라이버
Oracle 전용
프로토콜
Oracle
표준 API 사용

DBMS 종속 탈피



- **다양한 DBMS 지원**

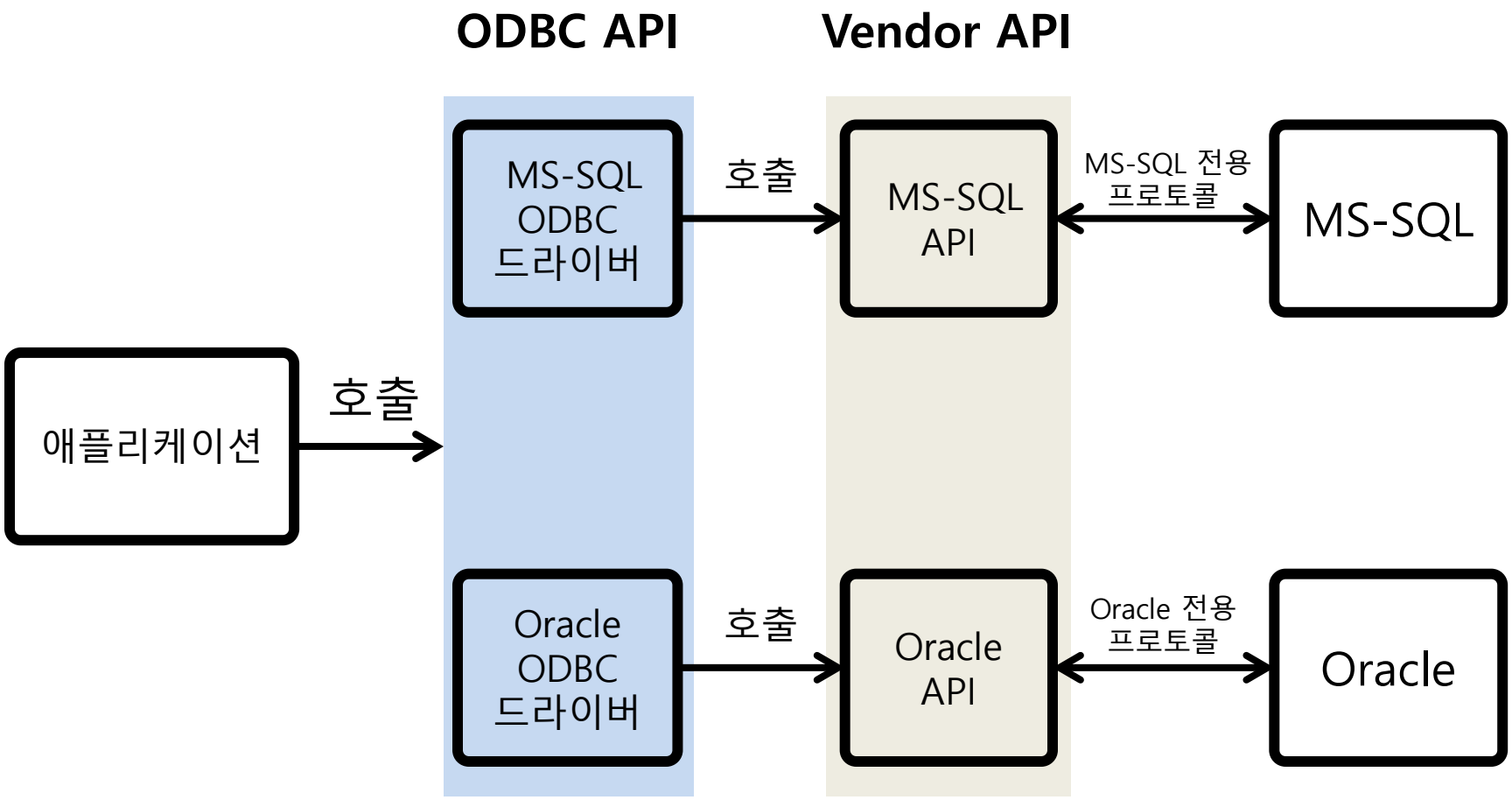
DBMS 종속 탈피



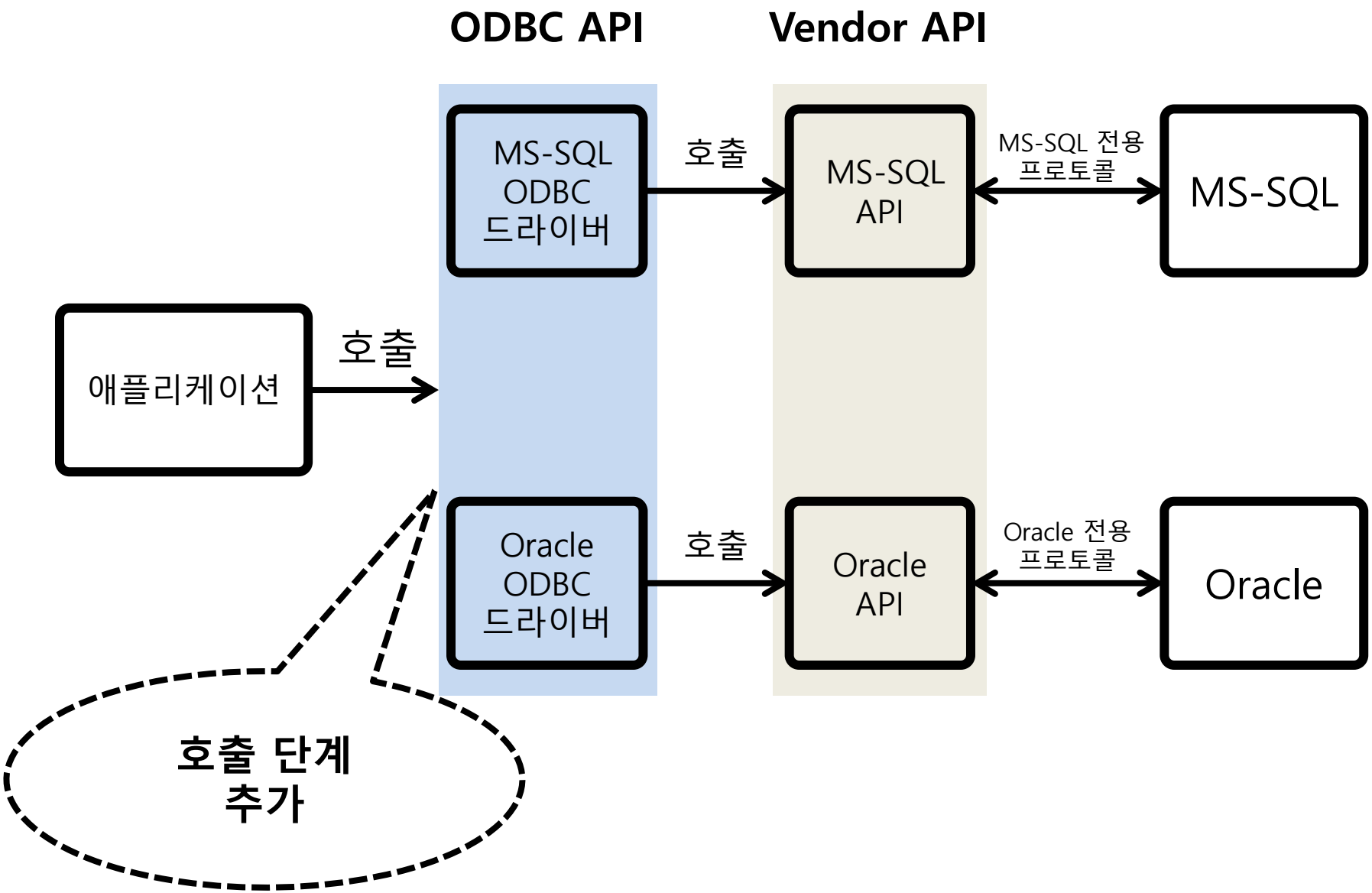
- **다양한 DBMS 지원**
- **개발비 절감**

ODBC 속도

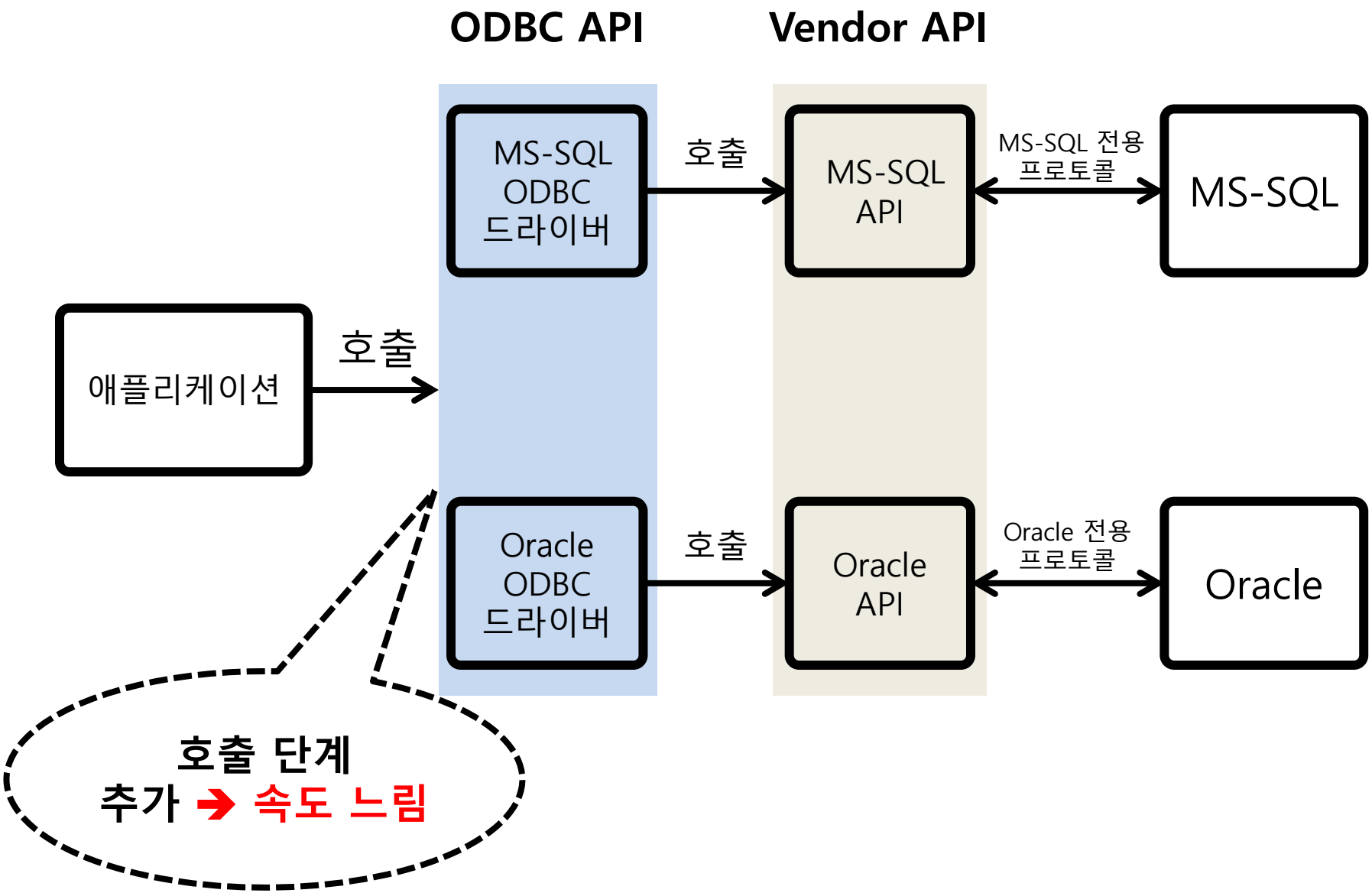
ODBC 속도


ODBC API
Vendor API
MS-SQL ODBC 드라이버
호출
MS-SQL API
MS-SQL 전용 프로토콜
MS-SQL
애플리케이션
호출
Oracle ODBC 드라이버
호출
Oracle API
Oracle 전용 프로토콜
Oracle

ODBC 속도

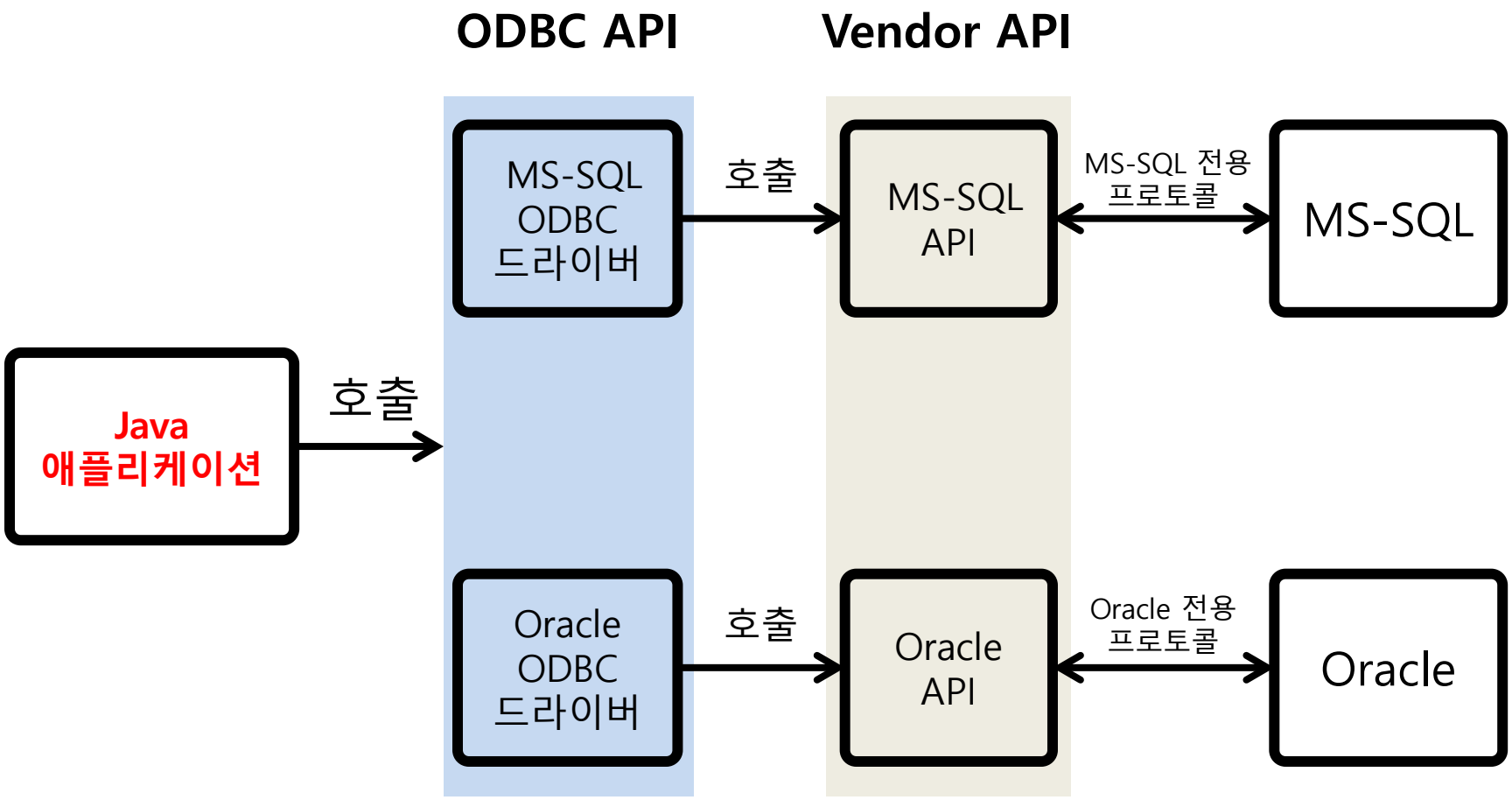

ODBC API
Vendor API
MS-SQL ODBC 드라이버
호출
MS-SQL API
MS-SQL 전용 프로토콜
MS-SQL
애플리케이션
호출
Oracle ODBC 드라이버
호출
Oracle API
Oracle 전용 프로토콜
Oracle
호출 단계 추가

ODBC 속도

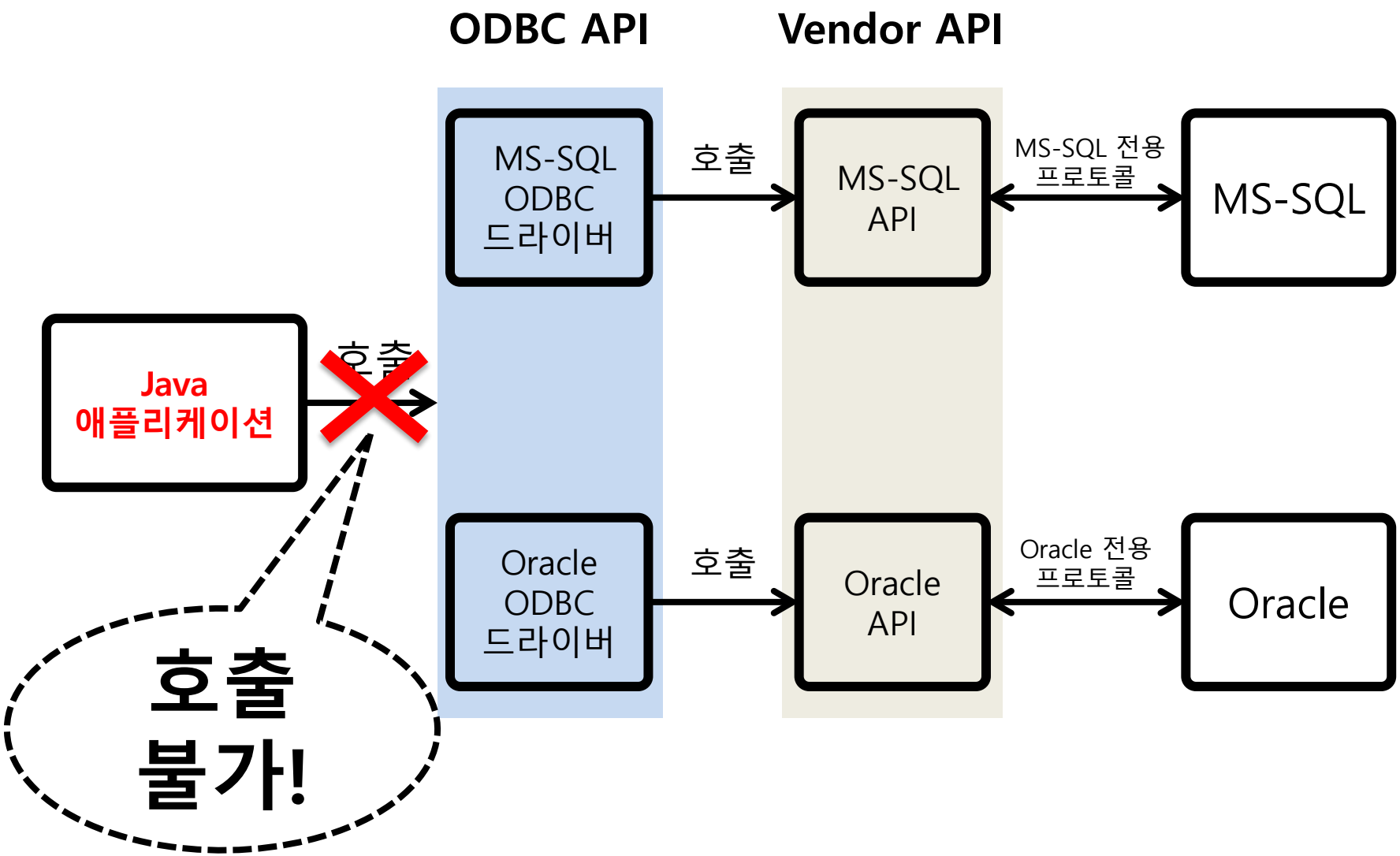

ODBC API
Vendor API
MS-SQL ODBC 드라이버
호출
MS-SQL API
MS-SQL 전용 프로토콜
MS-SQL
애플리케이션
호출
Oracle ODBC 드라이버
호출
Oracle API
Oracle 전용 프로토콜
Oracle
호출 단계
추가 ➔ 속도 느림

Java 애플리케이션에서 DBMS 접속

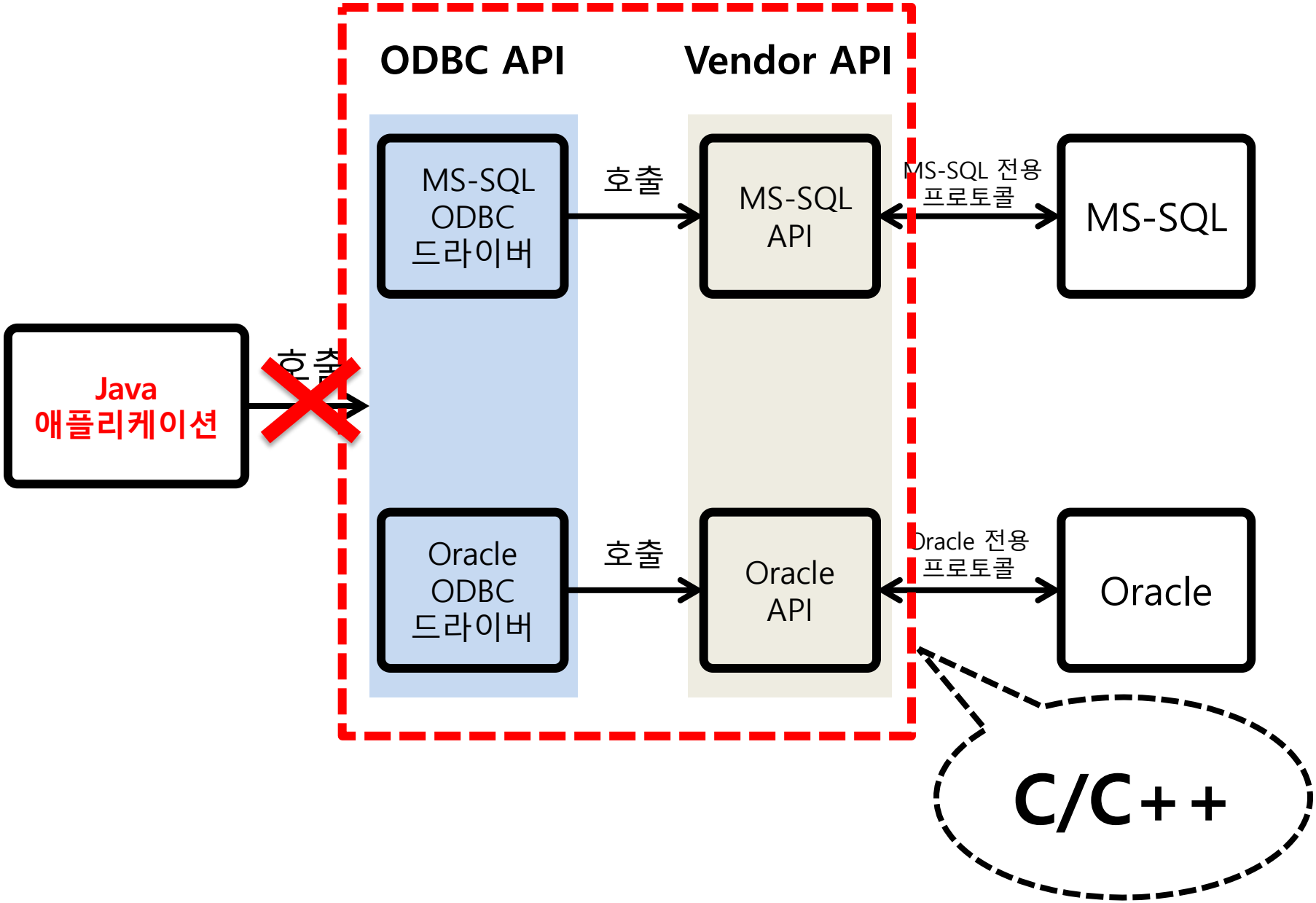
DBMS에 종속


ODBC API
Vendor API
MS-SQL ODBC 드라이버
호출
MS-SQL API
MS-SQL 전용 프로토콜
MS-SQL
Java 애플리케이션
호출
Oracle ODBC 드라이버
호출
Oracle API
Oracle 전용 프로토콜
Oracle

DBMS에 종속


ODBC API
Vendor API
MS-SQL ODBC 드라이버
호출
MS-SQL API
MS-SQL 전용 프로토콜
MS-SQL
Java 애플리케이션
호출
Oracle ODBC 드라이버
호출
Oracle API
Oracle 전용 프로토콜
Oracle
호출 불가!

DBMS에 종속


ODBC API
Vendor API
MS-SQL ODBC 드라이버
호출
MS-SQL API
MS-SQL 전용 프로토콜
MS-SQL
Java 애플리케이션
호출
Oracle ODBC 드라이버
호출
Oracle API
Oracle 전용 프로토콜
Oracle
C/C++

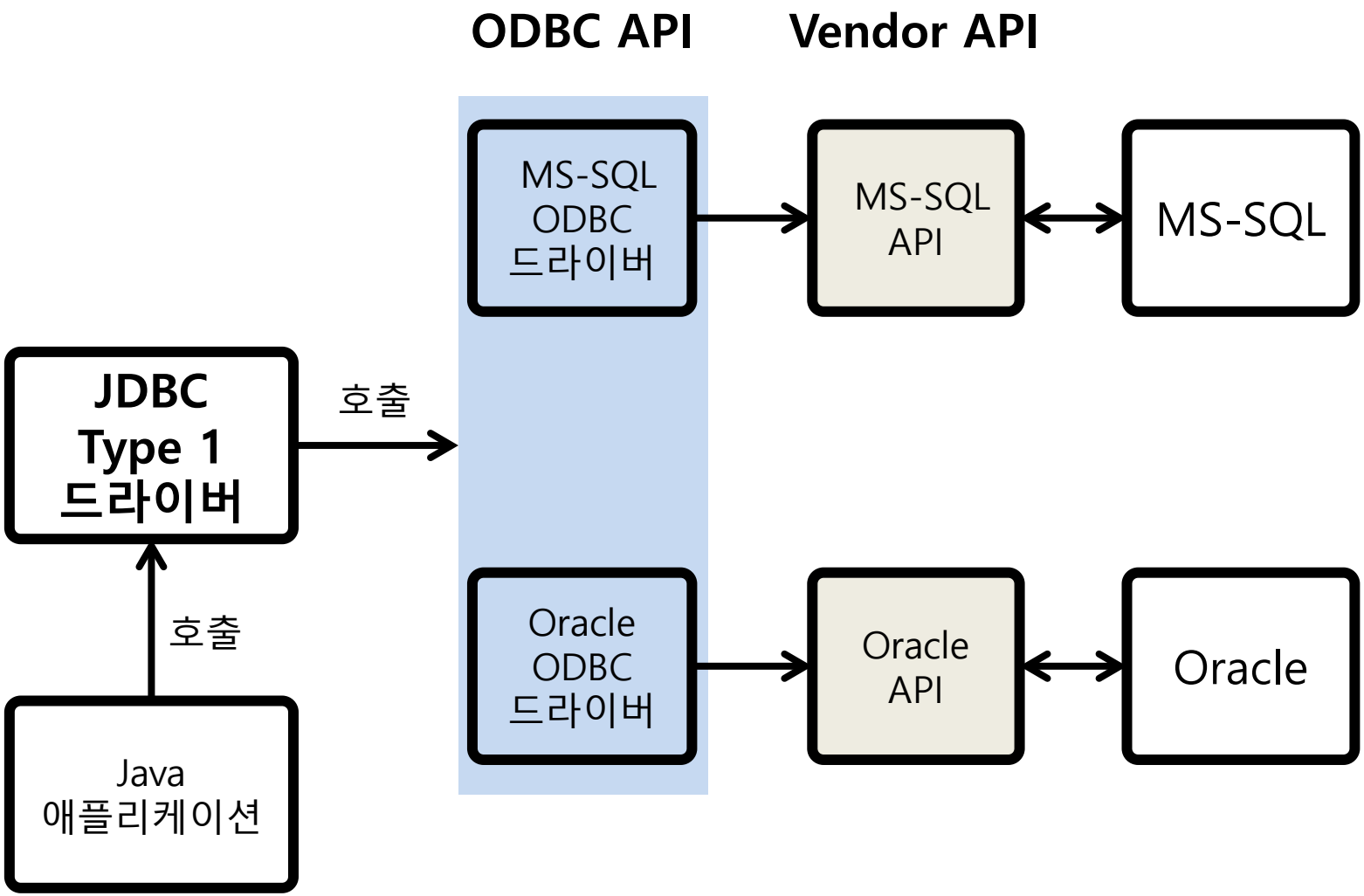
JDBC API

Java DataBase Connectivity

Java 애플리케이션을 위한 DBMS 접속 인터페이스

JDBC Type 1 드라이버

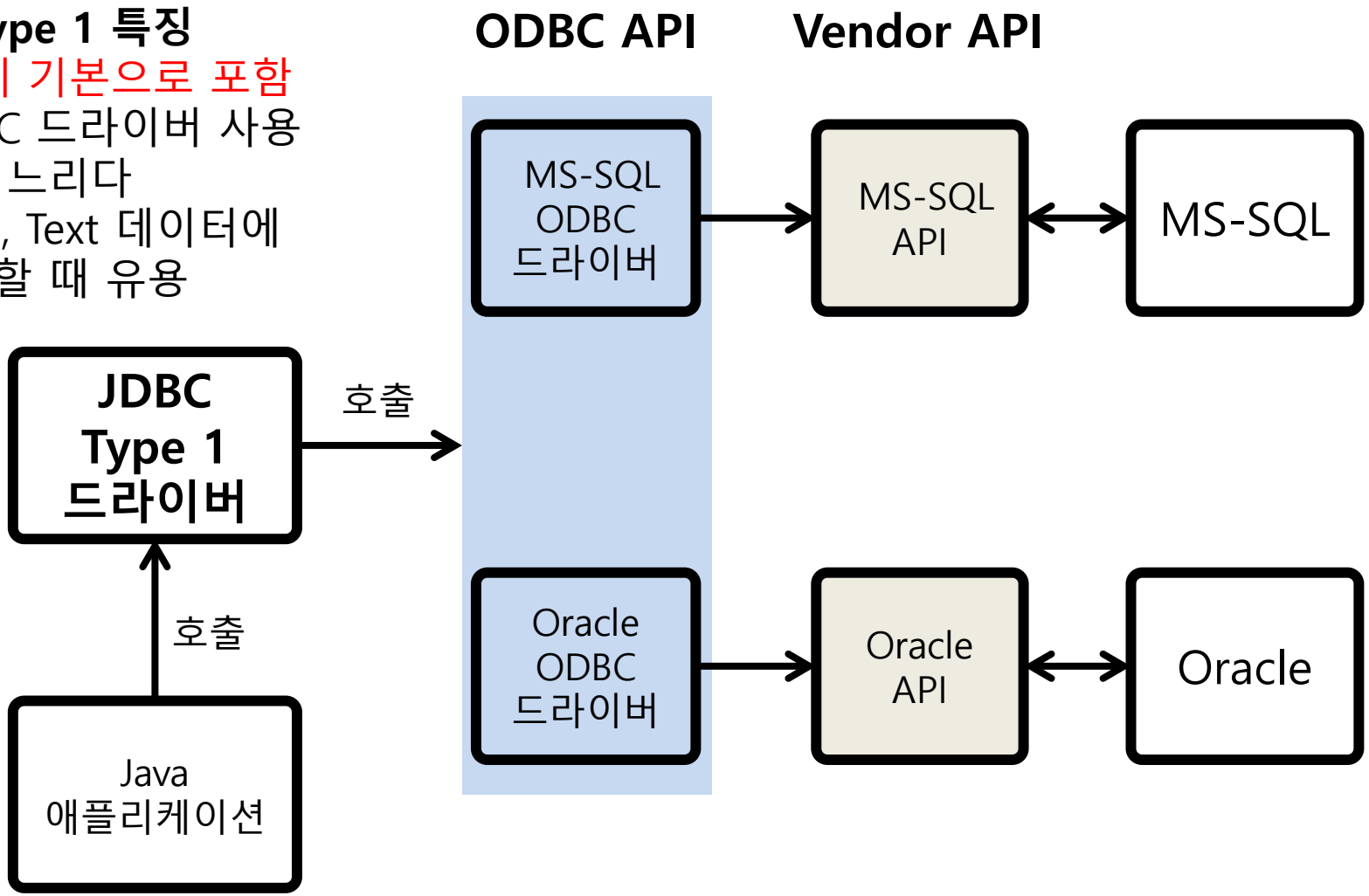
JDBC Type 1 드라이버



JDBC Type 1 드라이버

JDBC Type 1 특징

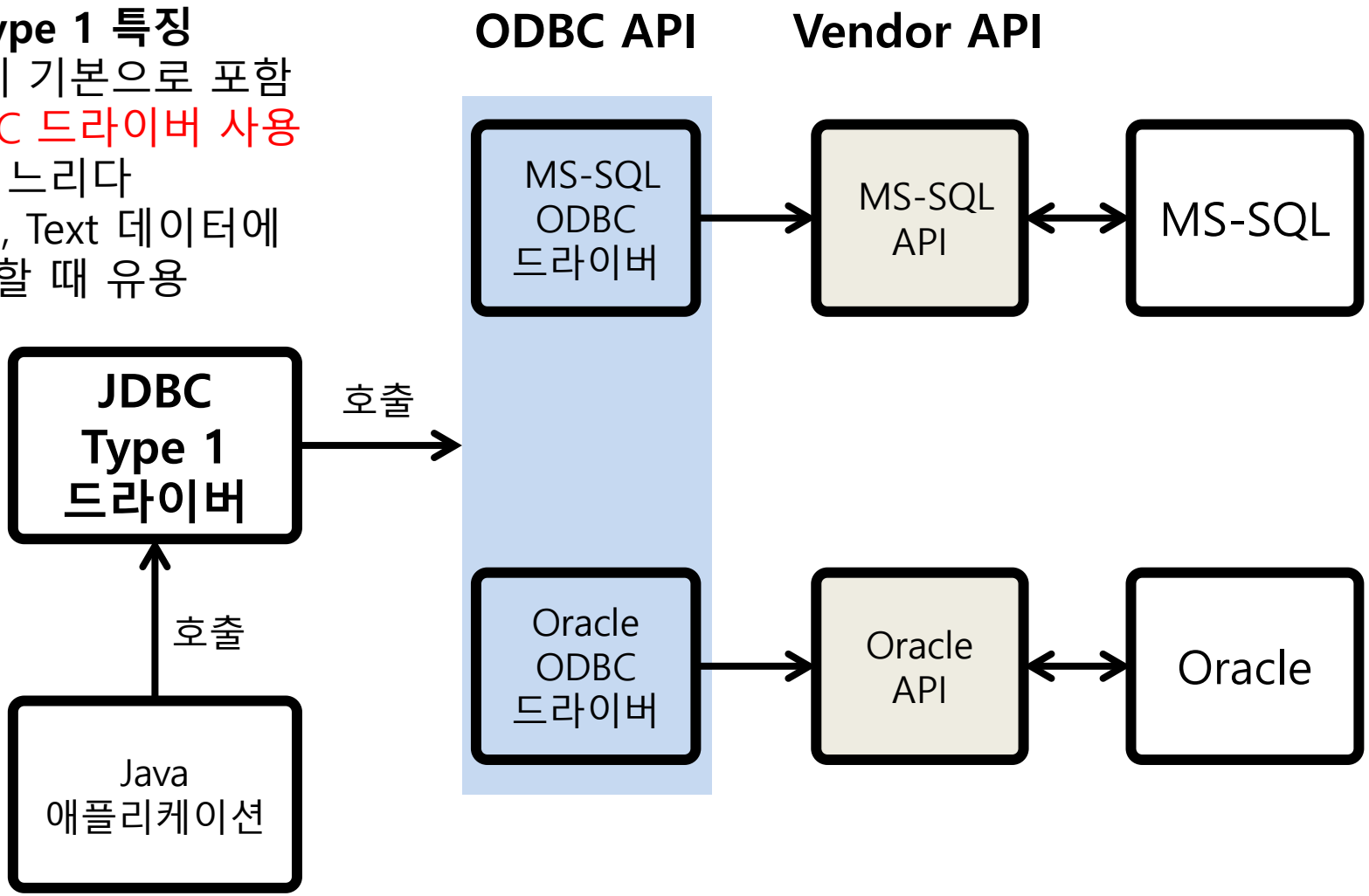
- JRE에 기본으로 포함
- ODBC 드라이버 사용
- 속도 느리다
- Excel, Text 데이터에 접근할 때 유용



JDBC Type 1 드라이버

JDBC Type 1 특징

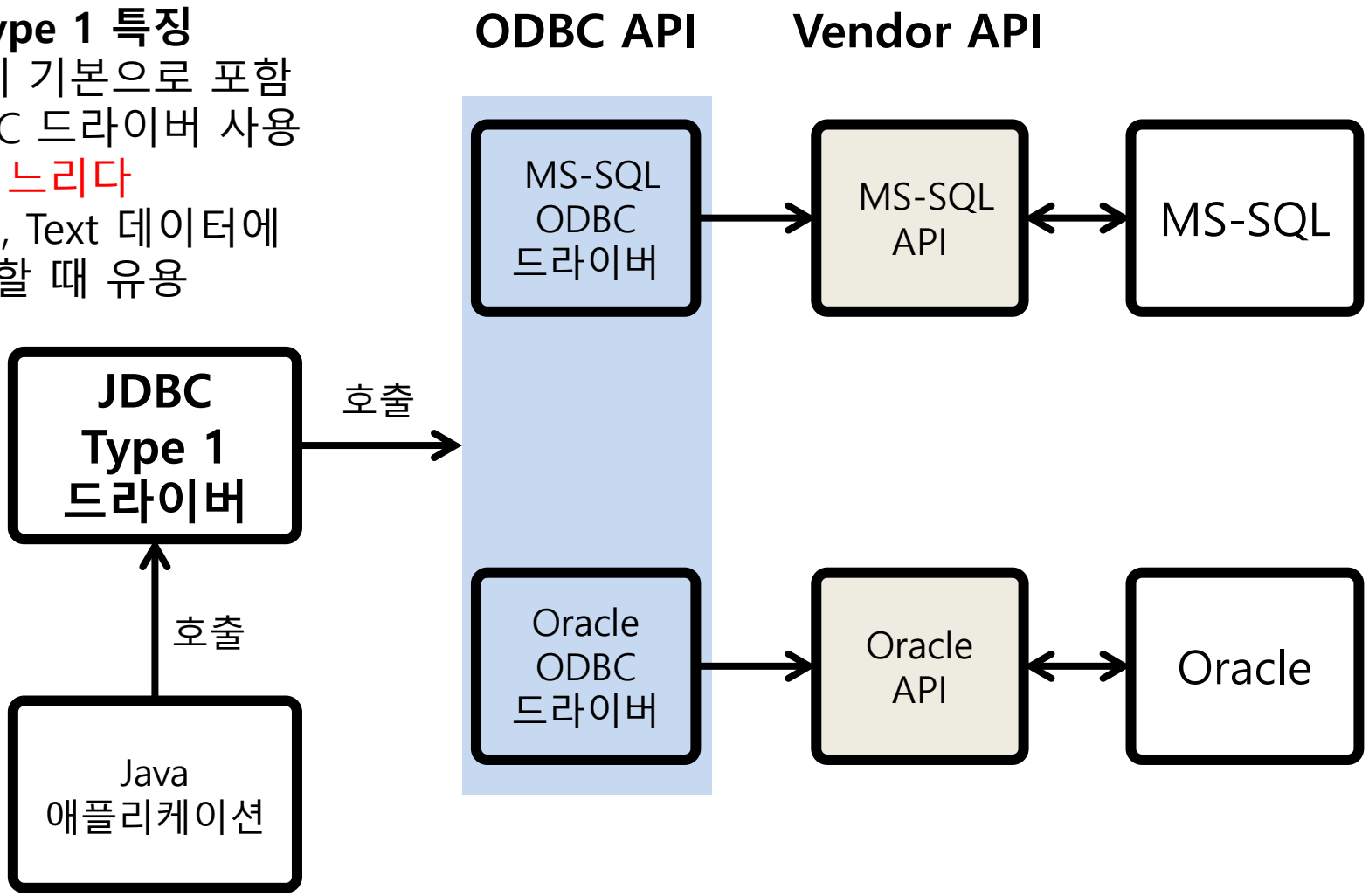
- JRE에 기본으로 포함
- ODBC 드라이버 사용
- 속도 느리다
- Excel, Text 데이터에 접근할 때 유용



JDBC Type 1 드라이버

JDBC Type 1 특징

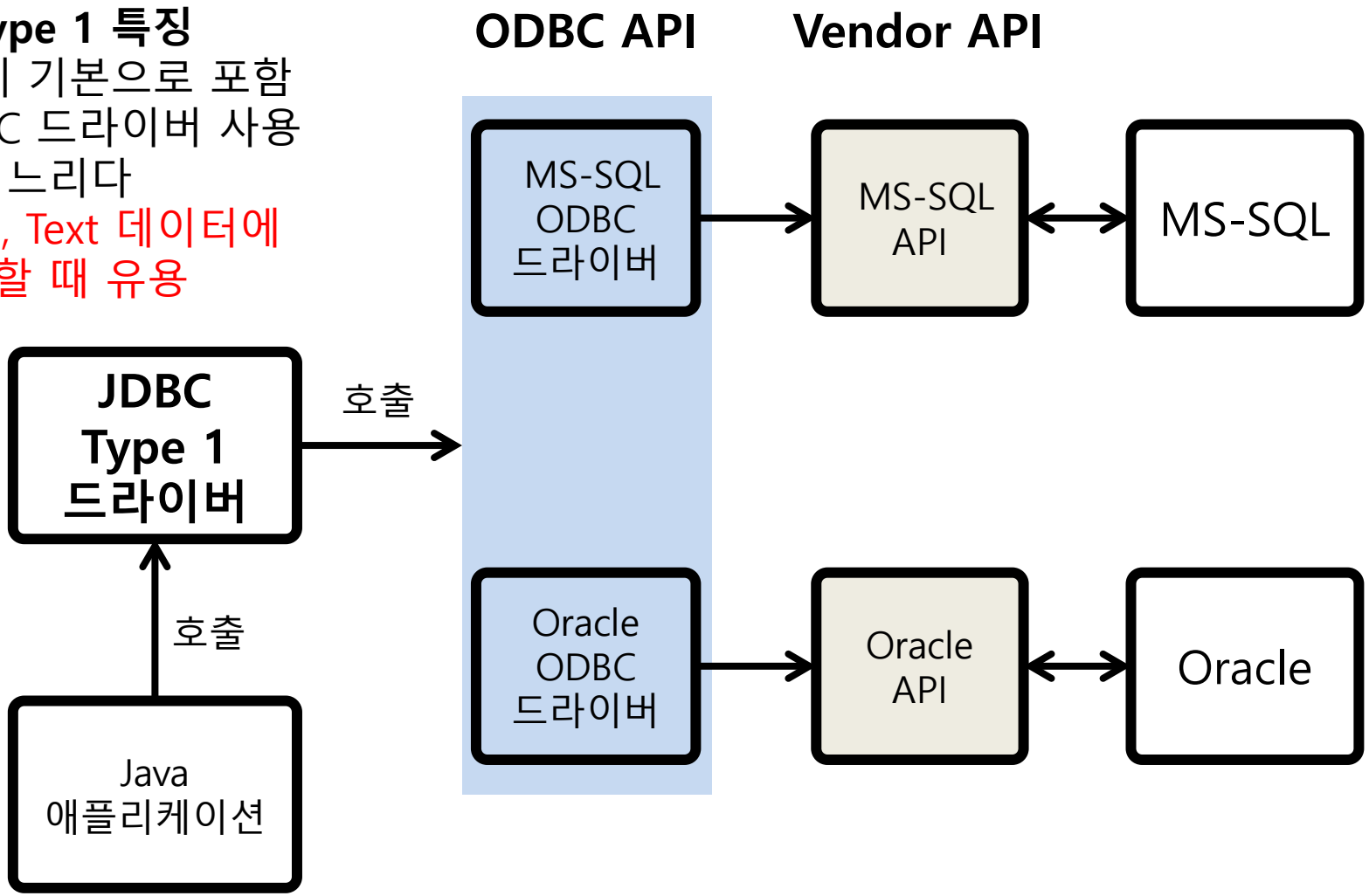
- JRE에 기본으로 포함
- ODBC 드라이버 사용
- 속도 느리다
- Excel, Text 데이터에 접근할 때 유용



JDBC Type 1 드라이버

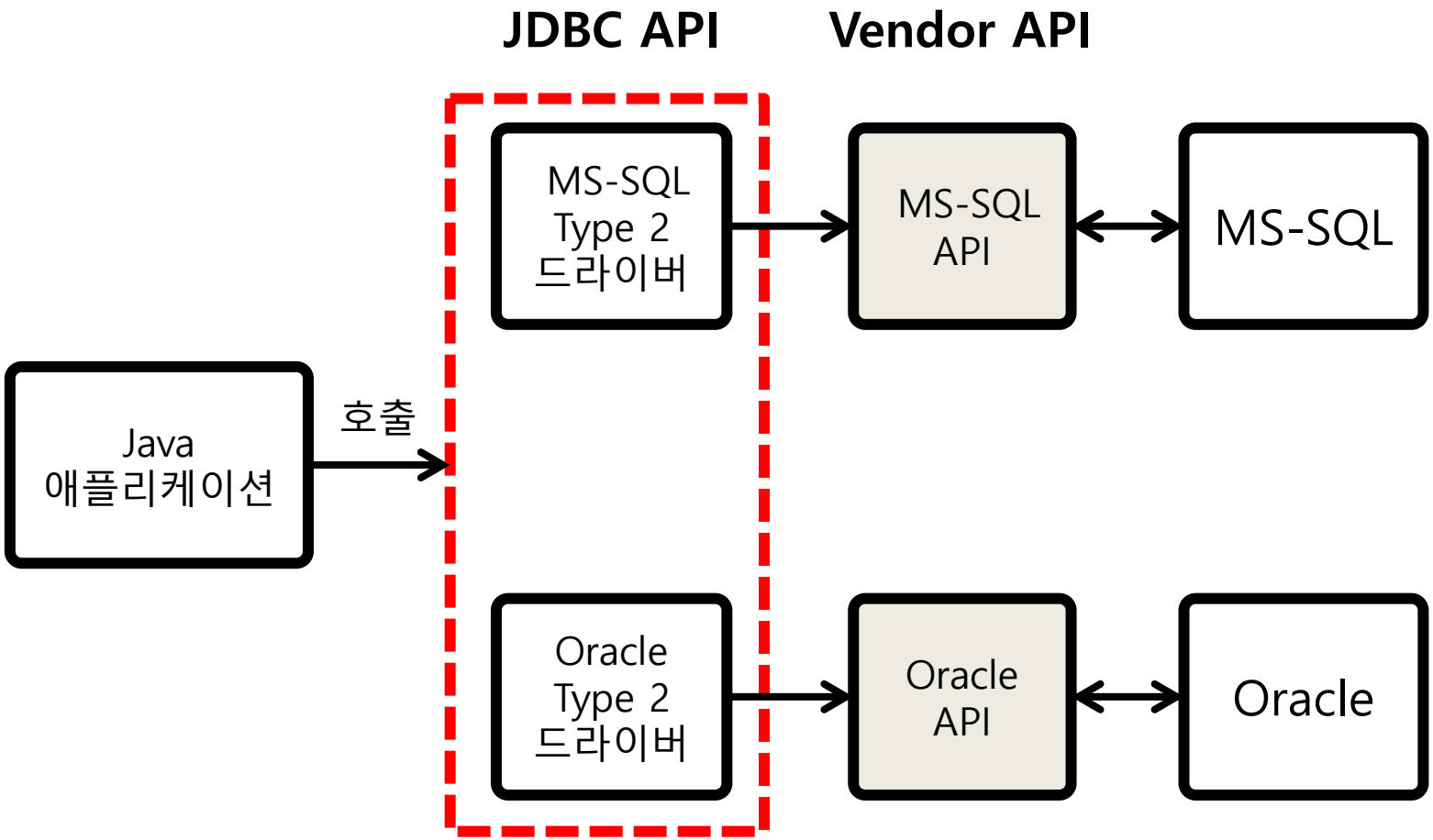
JDBC Type 1 특징

- JRE에 기본으로 포함
- ODBC 드라이버 사용
- 속도 느리다
- Excel, Text 데이터에 접근할 때 유용

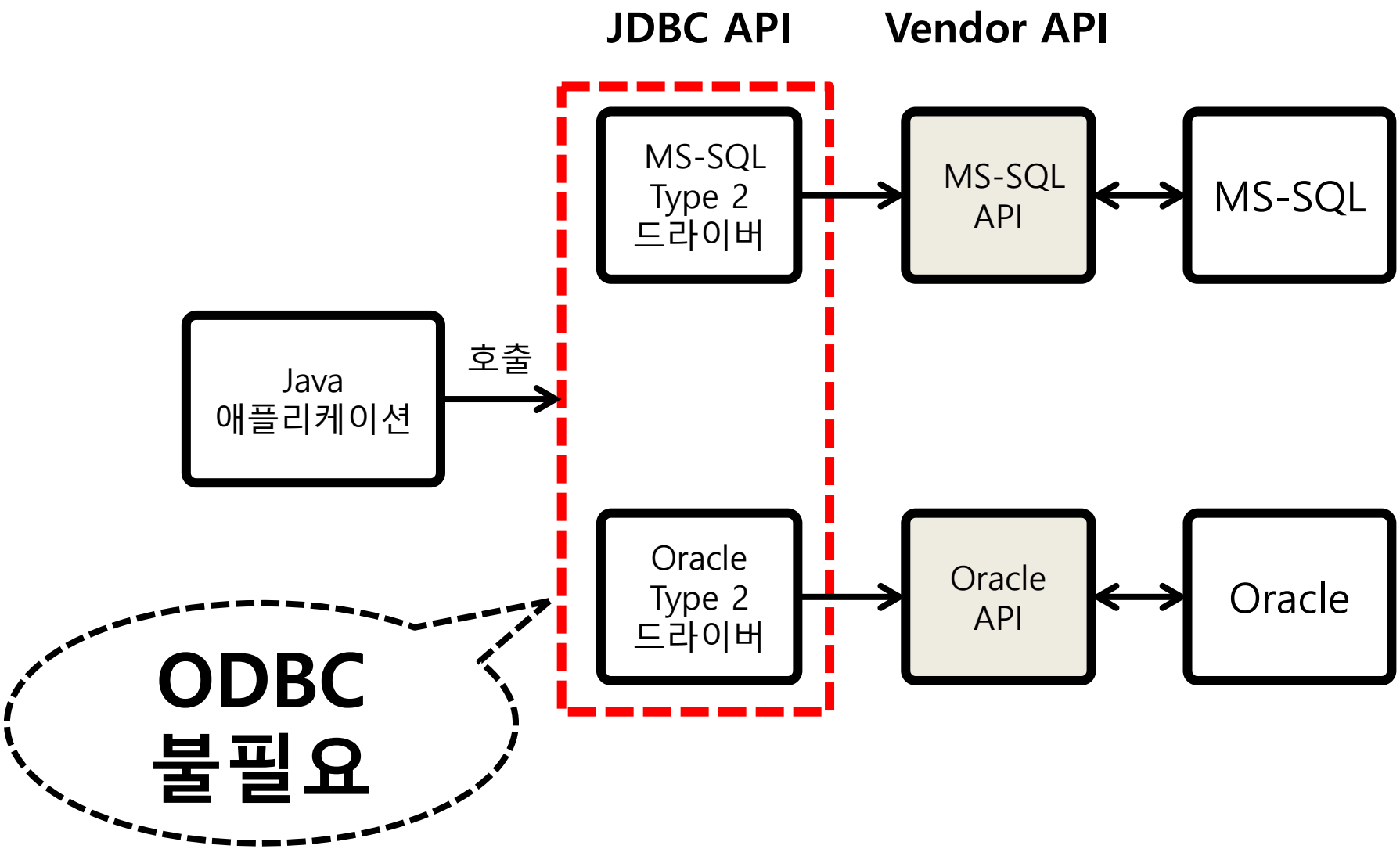


JDBC Type 2 드라이버

JDBC Type 2 드라이버



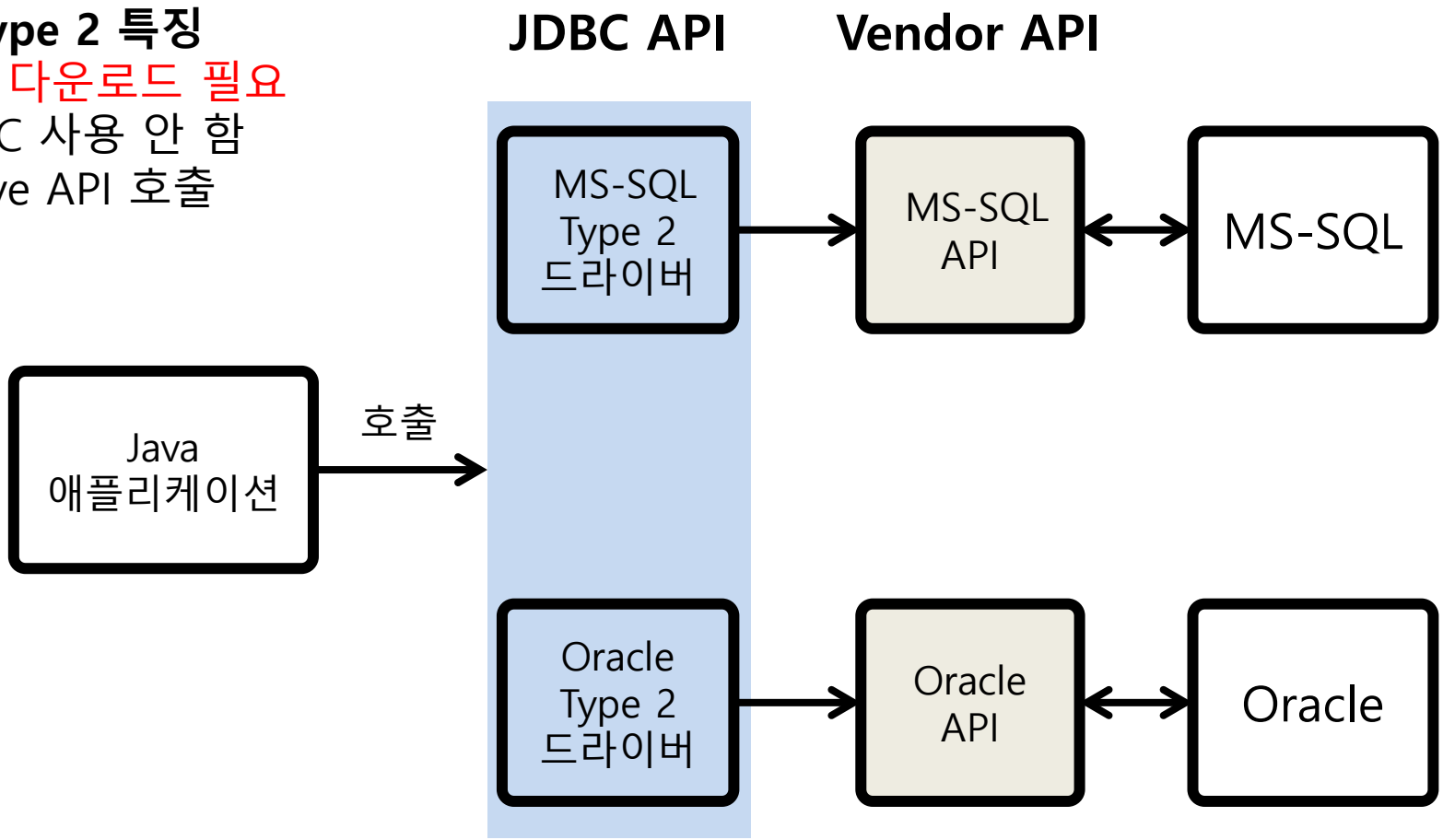
JDBC Type 2 드라이버



JDBC Type 2 드라이버

JDBC Type 2 특징

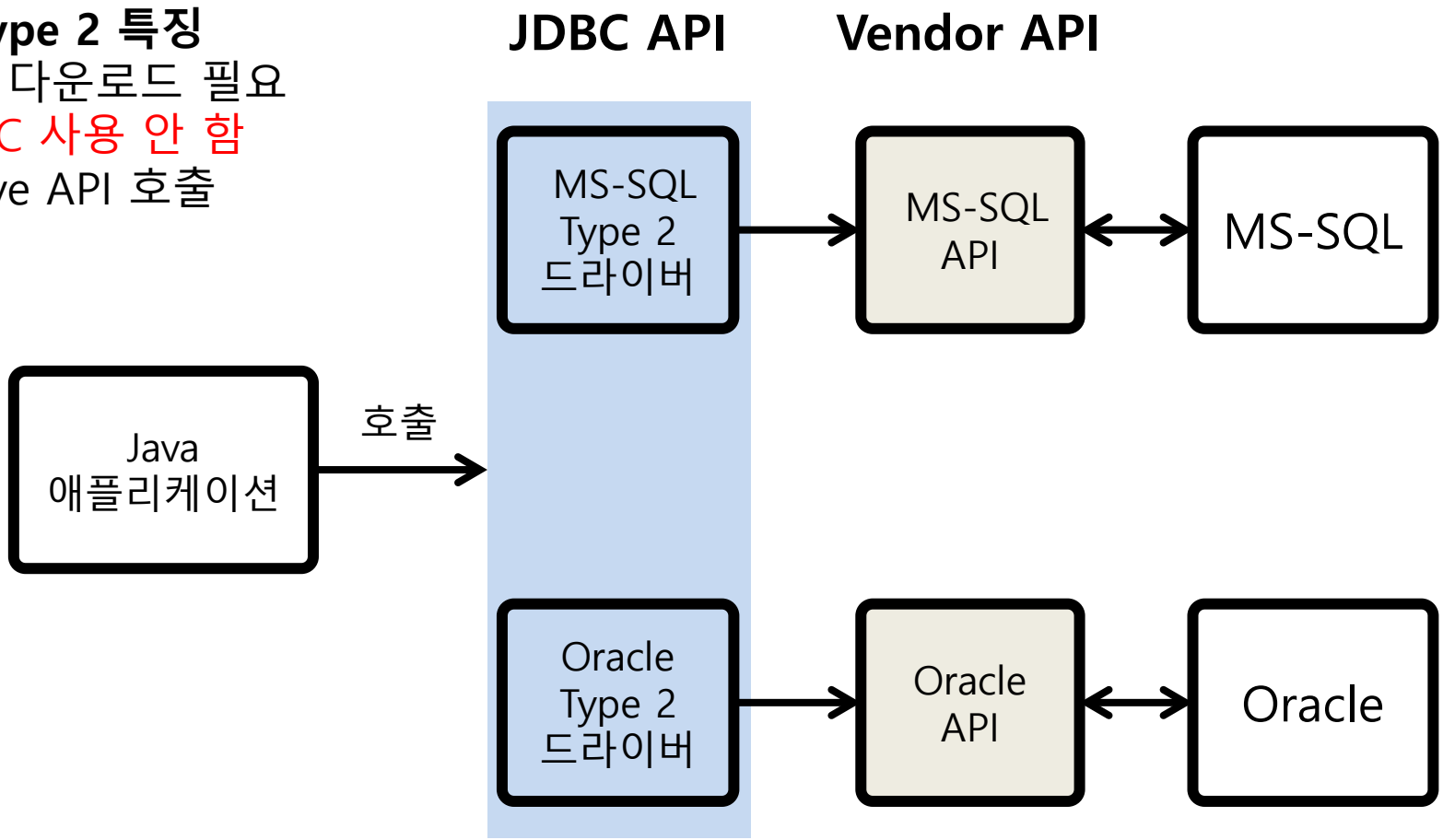
- 별도 다운로드 필요
- ODBC 사용 안 함
- Native API 호출



JDBC Type 2 드라이버

JDBC Type 2 특징

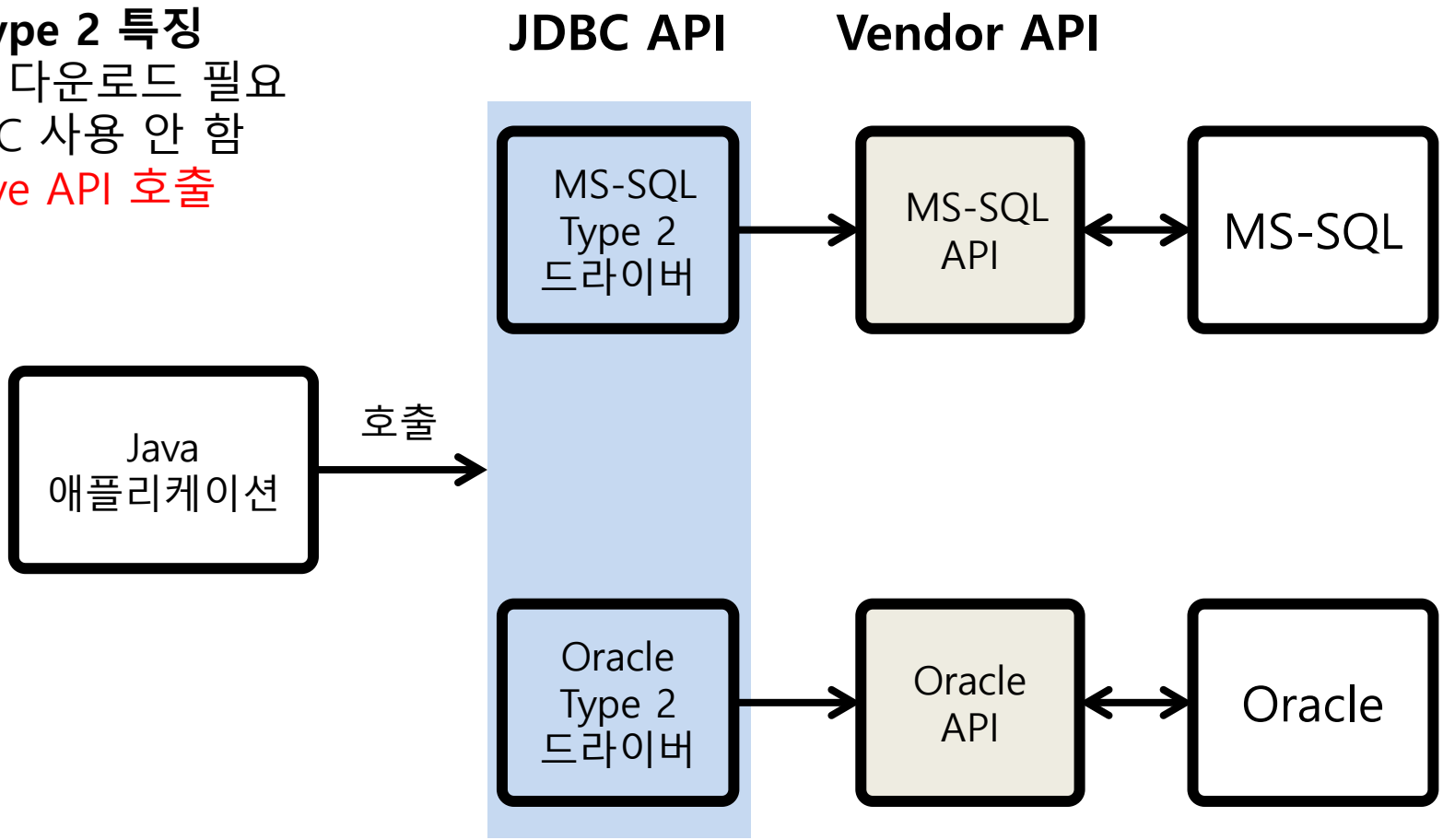
- 별도 다운로드 필요
- ODBC 사용 안 함
- Native API 호출



JDBC Type 2 드라이버

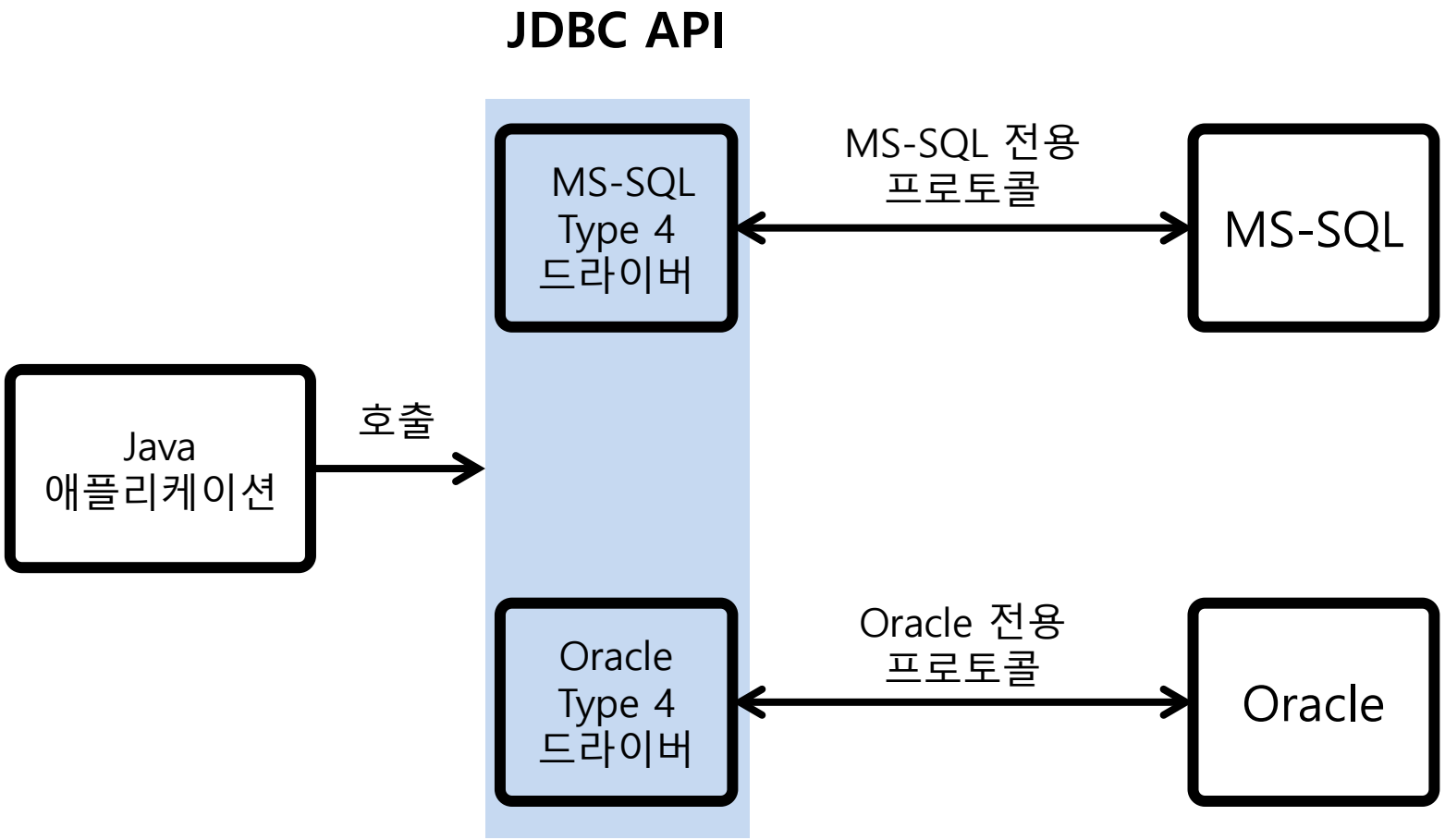
JDBC Type 2 특징

- 별도 다운로드 필요
- ODBC 사용 안 함
- Native API 호출



JDBC Type 4 드라이버

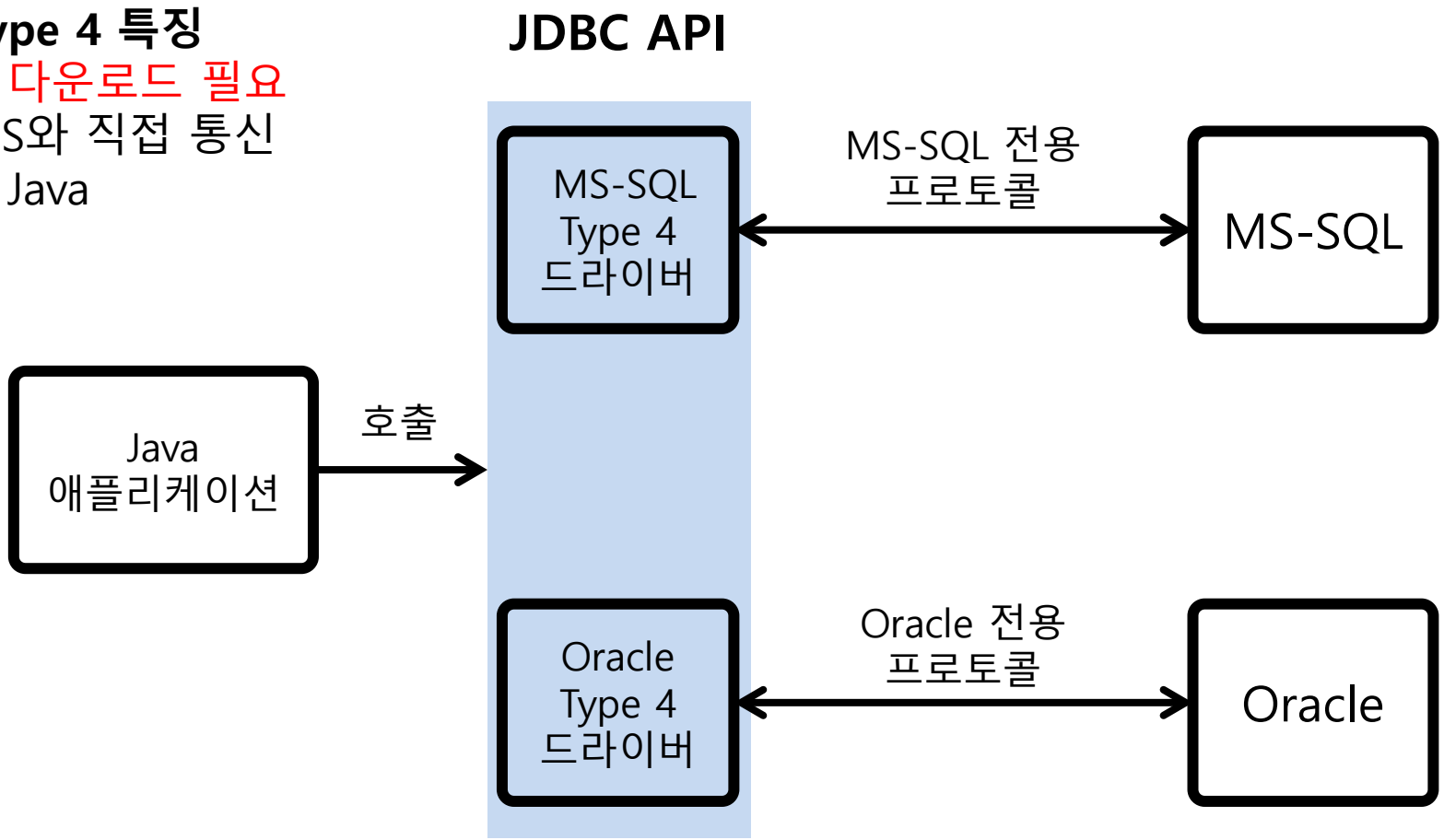
JDBC Type 4 드라이버



JDBC Type 4 드라이버

JDBC Type 4 특징

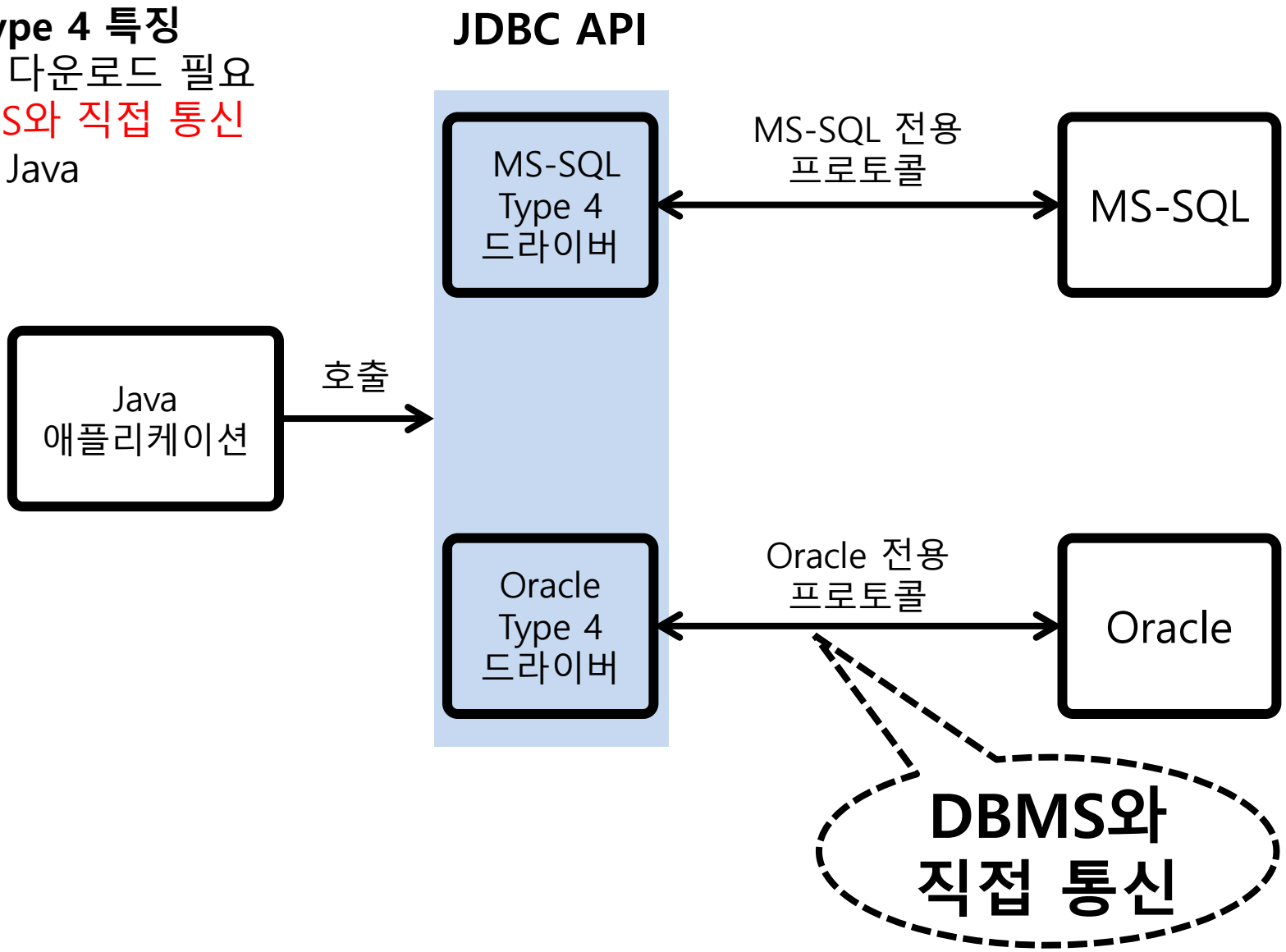
- 별도 다운로드 필요
- DBMS와 직접 통신
- Pure Java



JDBC Type 4 드라이버

JDBC Type 4 특징

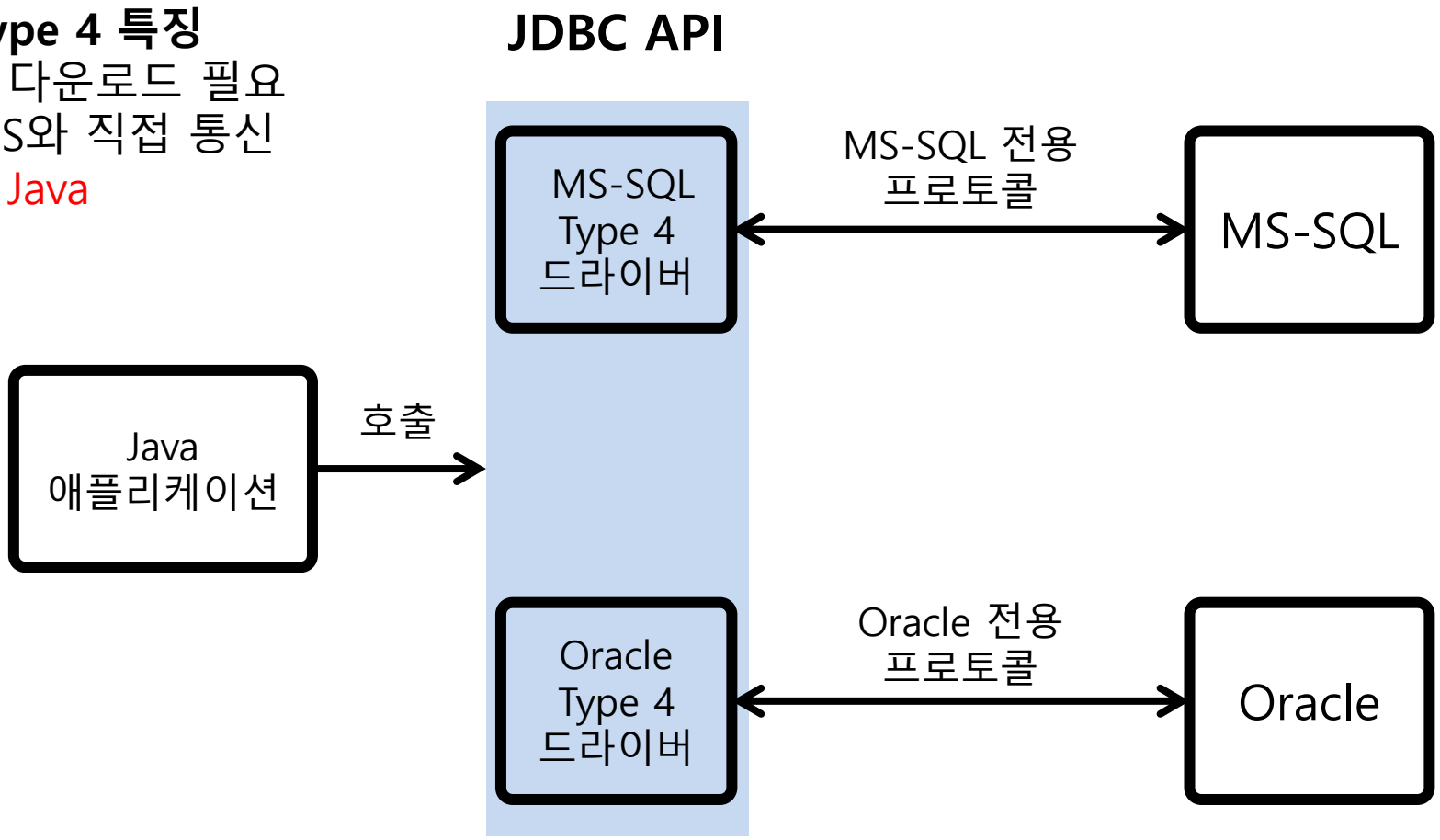
- 별도 다운로드 필요
- DBMS와 직접 통신
- Pure Java



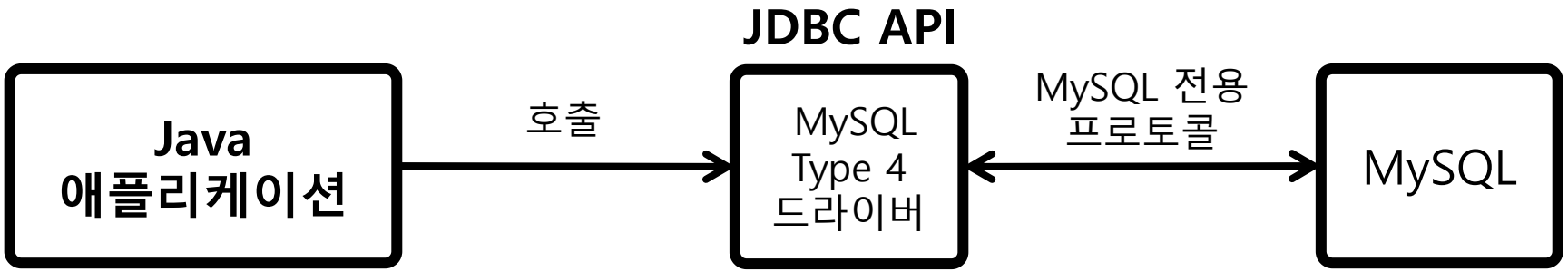
JDBC Type 4 드라이버

JDBC Type 4 특징

- 별도 다운로드 필요
- DBMS와 직접 통신
- Pure Java



실습 예제 아키텍처



4.2 HttpServlet으로 GET 요청 다루기

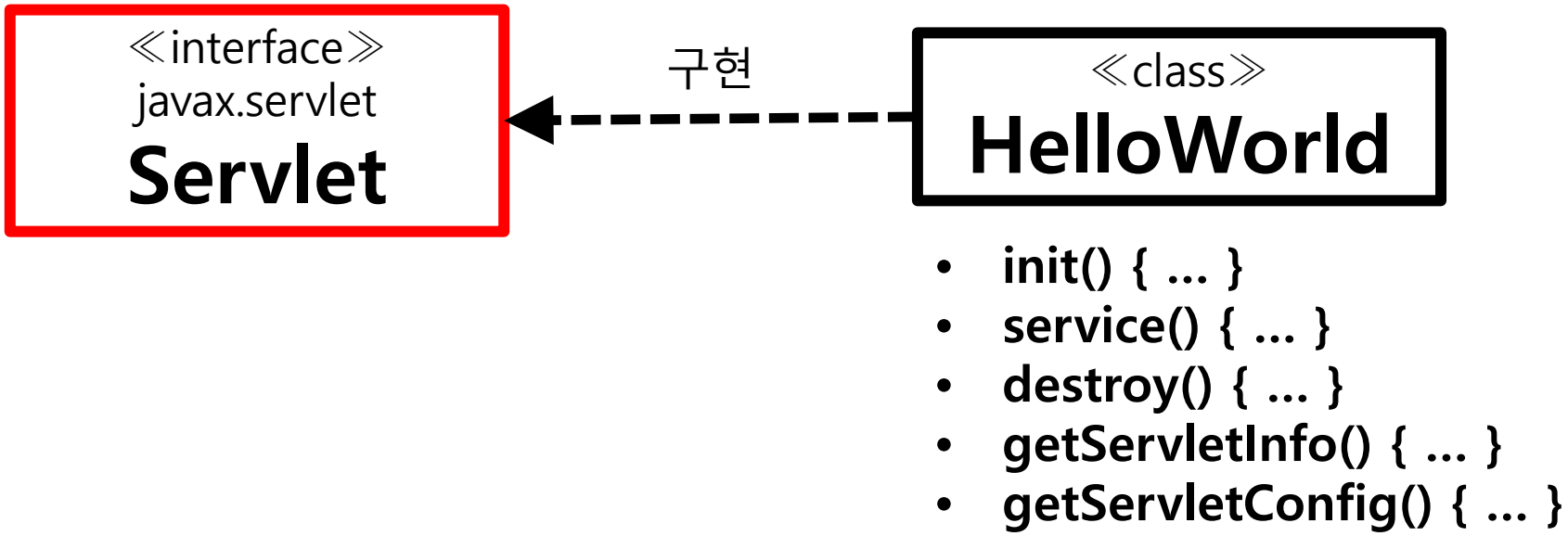
HttpServlet

서블릿 만들기

≪interface≫
javax.servlet
Servlet

서블릿 만들기

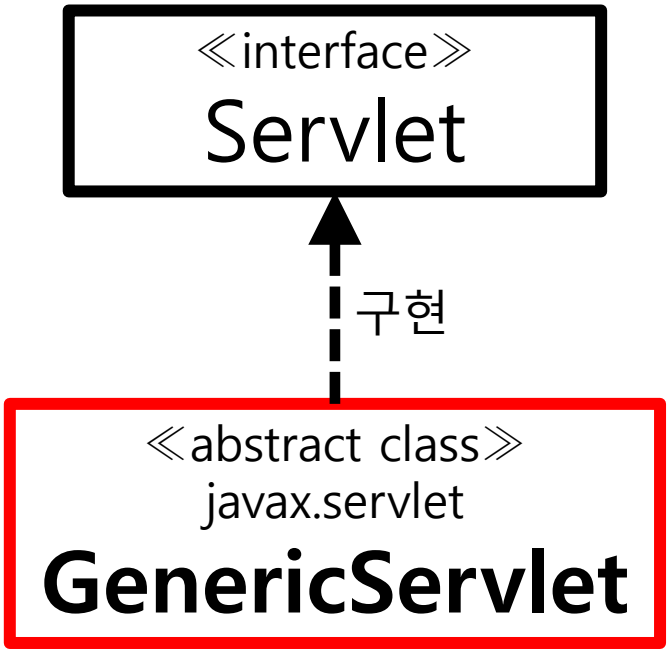
서블릿 만들기


≪interface≫
javax.servlet
Servlet
구현
≪class≫
HelloWorld
• init() { … }
• service() { … }
• destroy() { … }
• getServletInfo() { … }
• getServletConfig() { … }

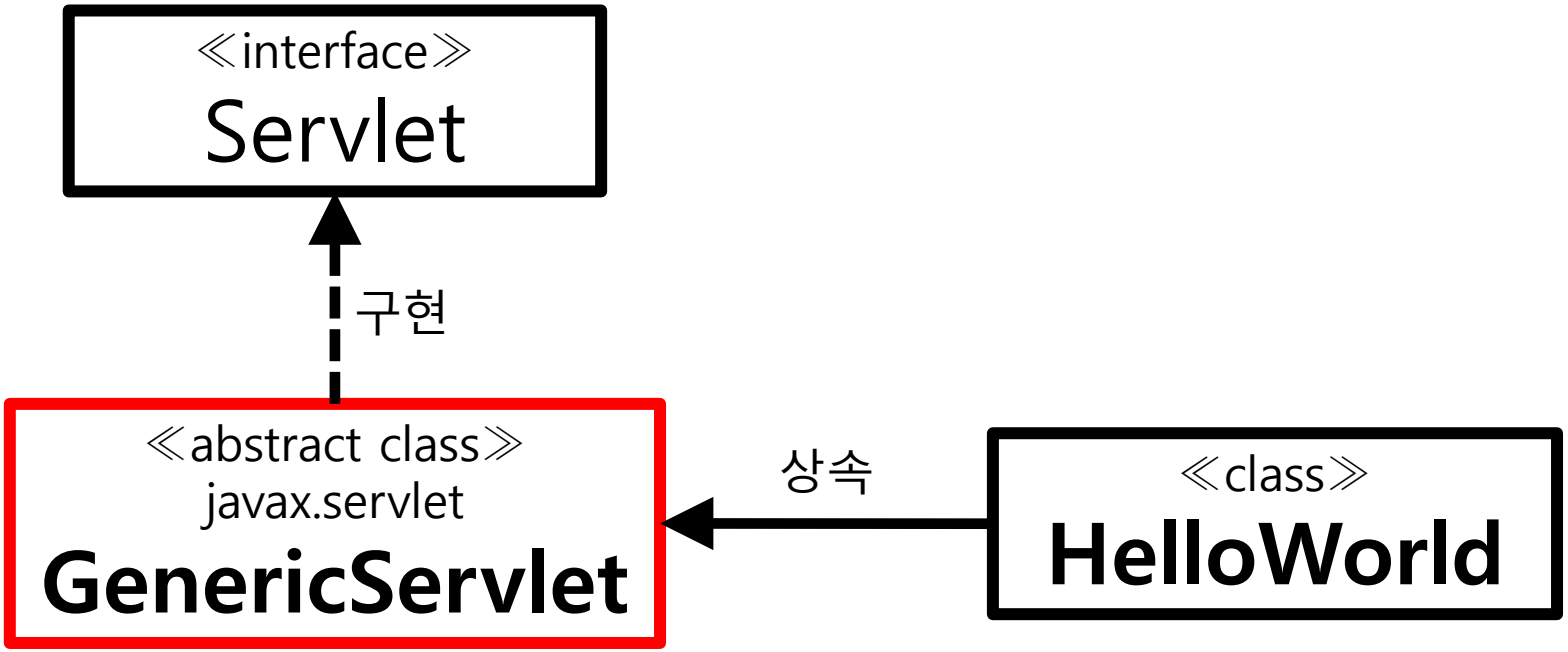
서블릿 만들기

≪interface≫
Servlet

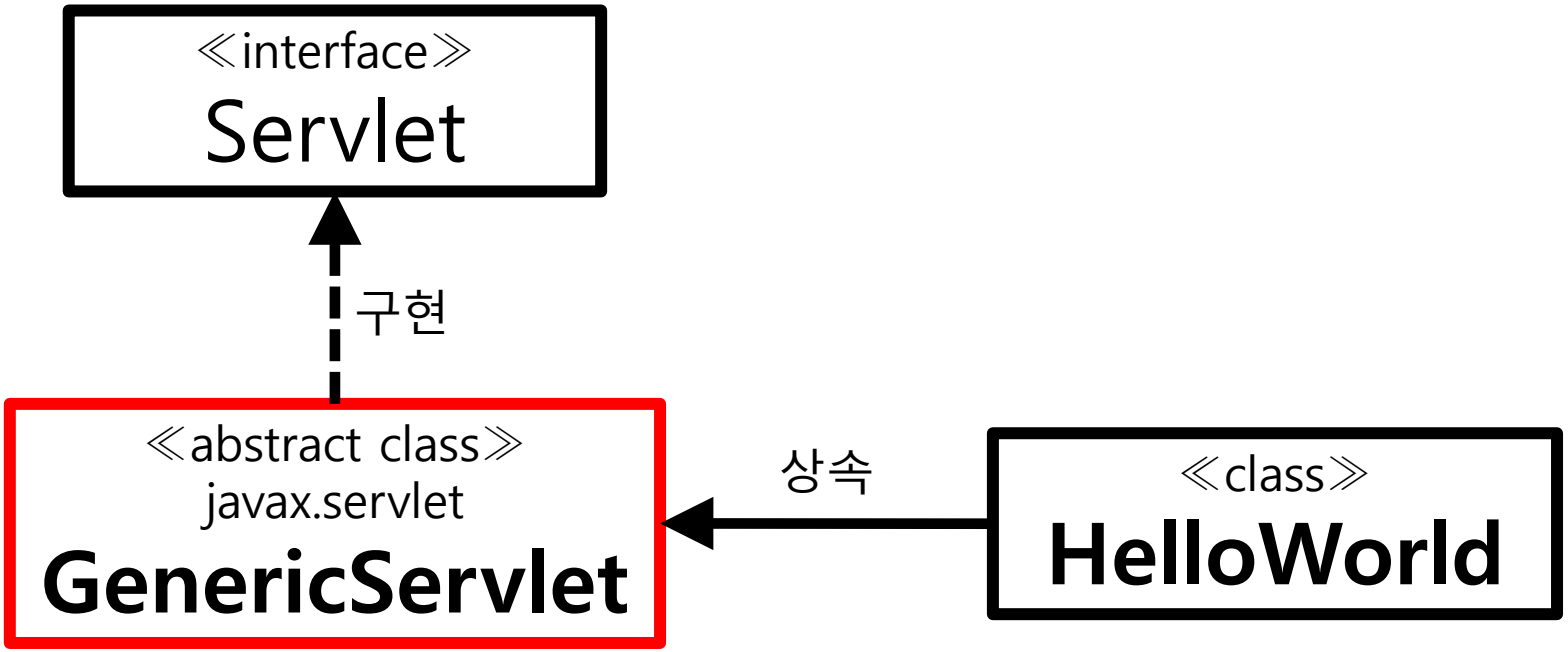
서블릿 만들기


≪interface≫
Servlet
구현
≪abstract class≫
javax.servlet
GenericServlet

서블릿 만들기

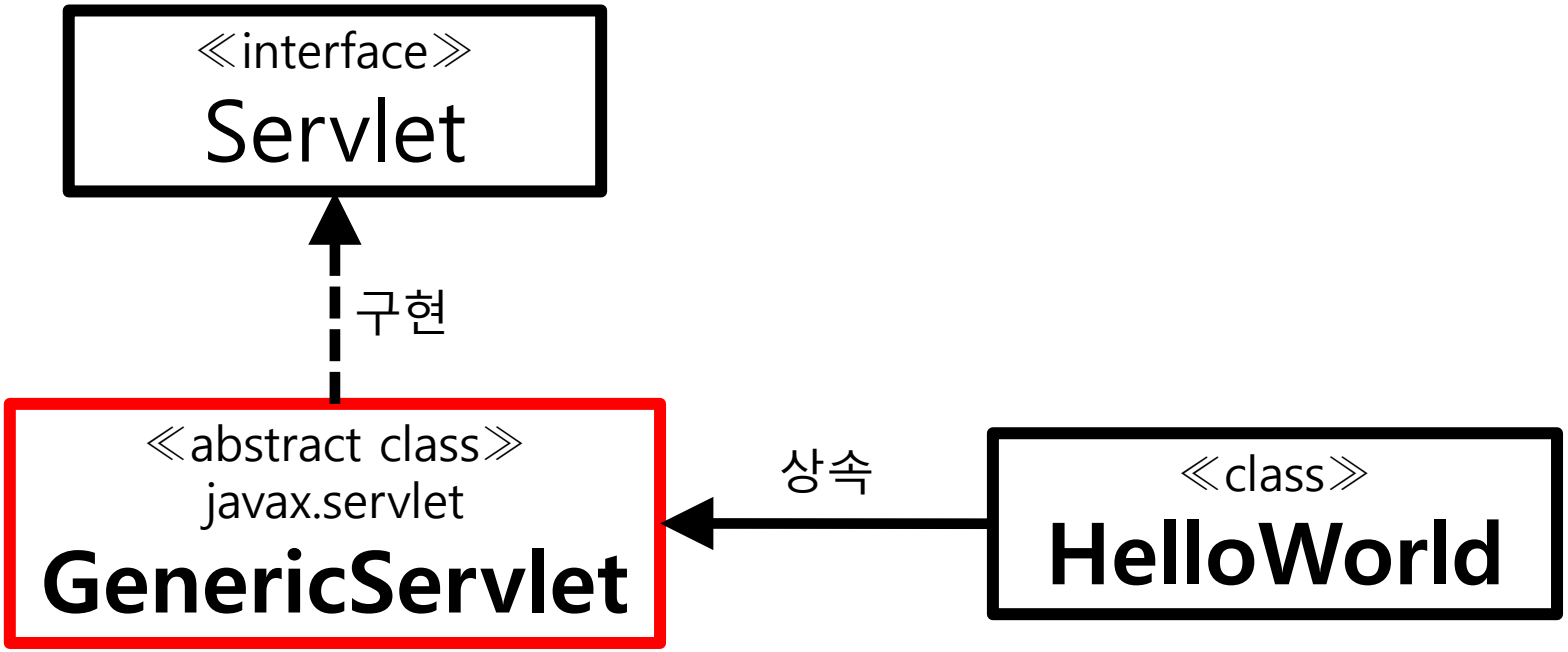

≪interface≫
Servlet
구현
≪abstract class≫
javax.servlet
GenericServlet
상속
≪class≫
HelloWorld

서블릿 만들기



- init() { … }
- destroy() { … }
- getServletInfo() { … }
- getServletConfig() { … }

서블릿 만들기



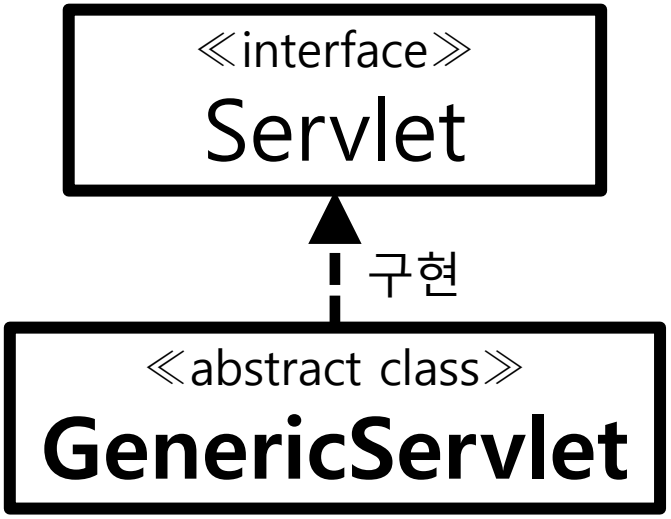
- init() { … }
- destroy() { … }
- getServletInfo() { … }
- getServletConfig() { … }

- **service() { … }**

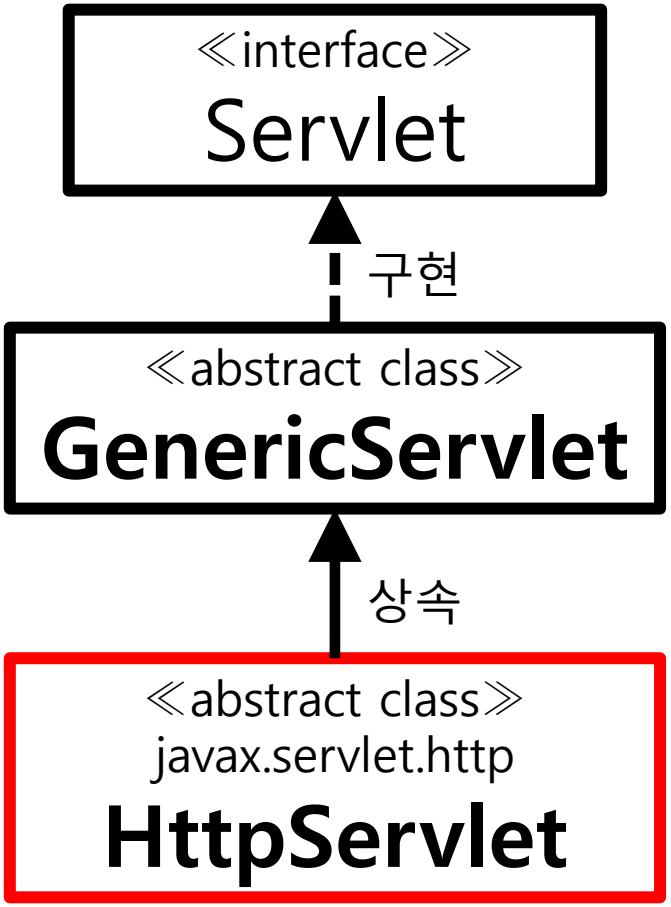
서블릿 만들기

≪interface≫
Servlet

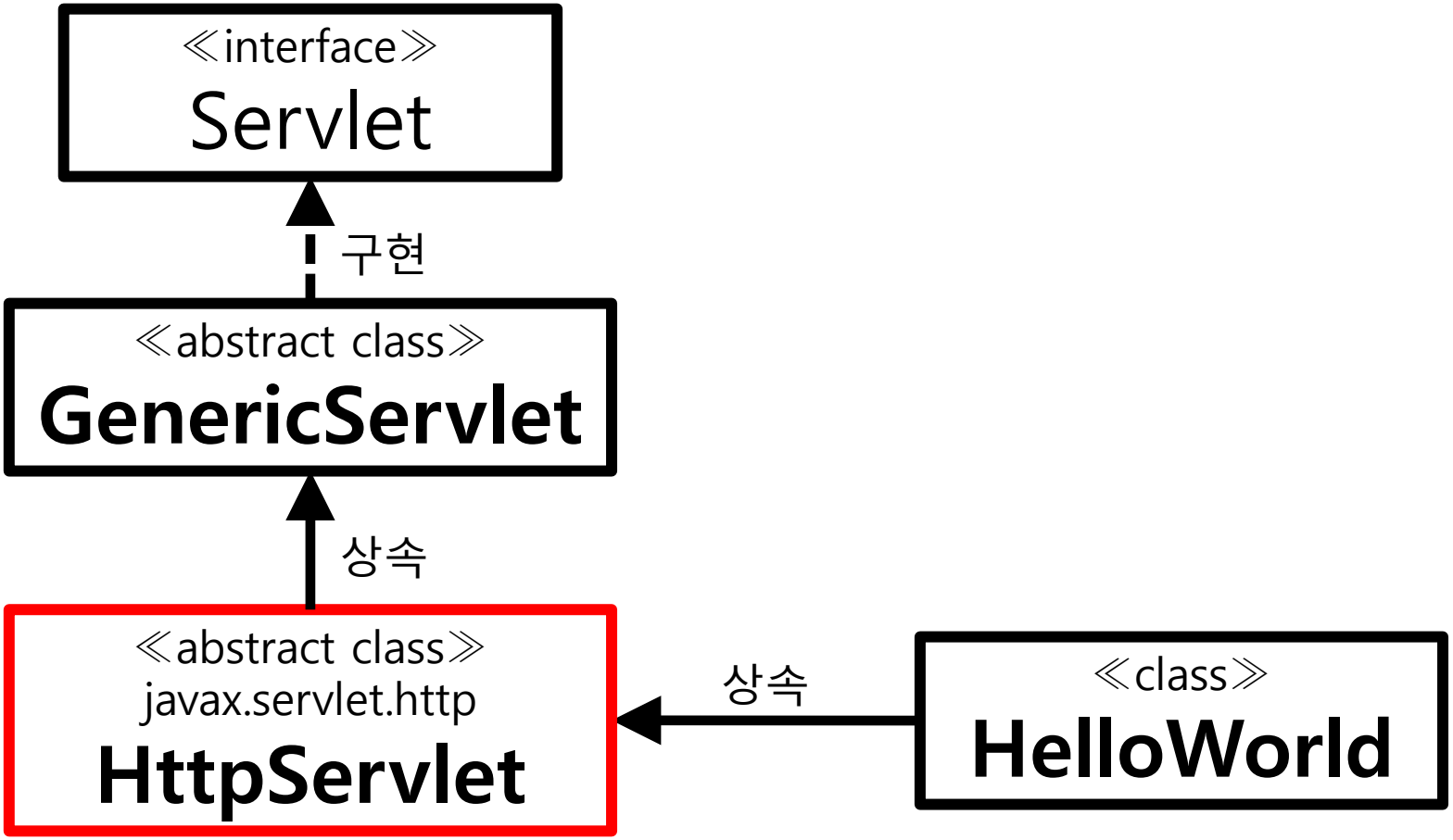
서블릿 만들기



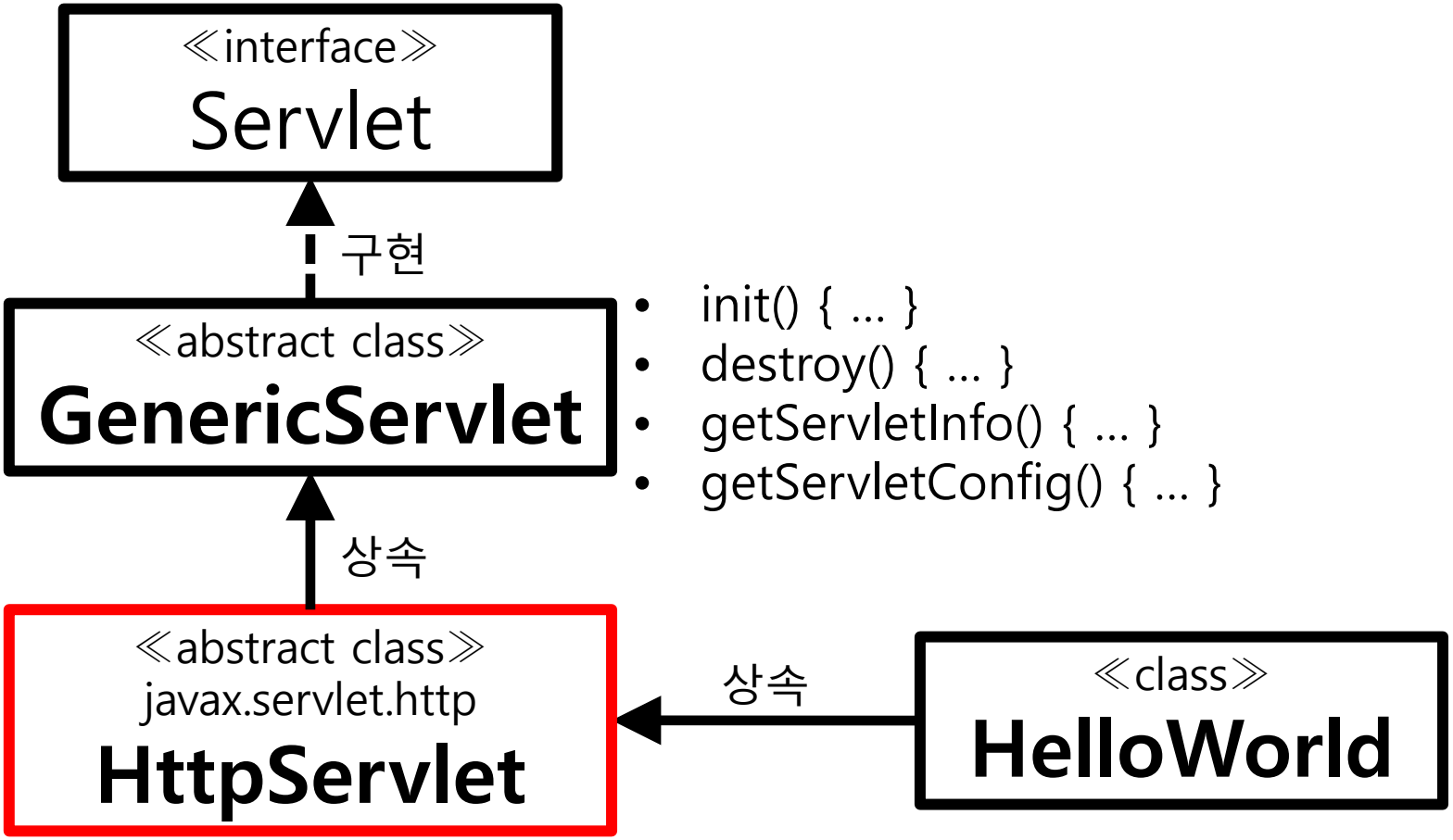
서블릿 만들기


≪interface≫
Servlet
구현
≪abstract class≫
GenericServlet
상속
≪abstract class≫
javax.servlet.http
HttpServlet

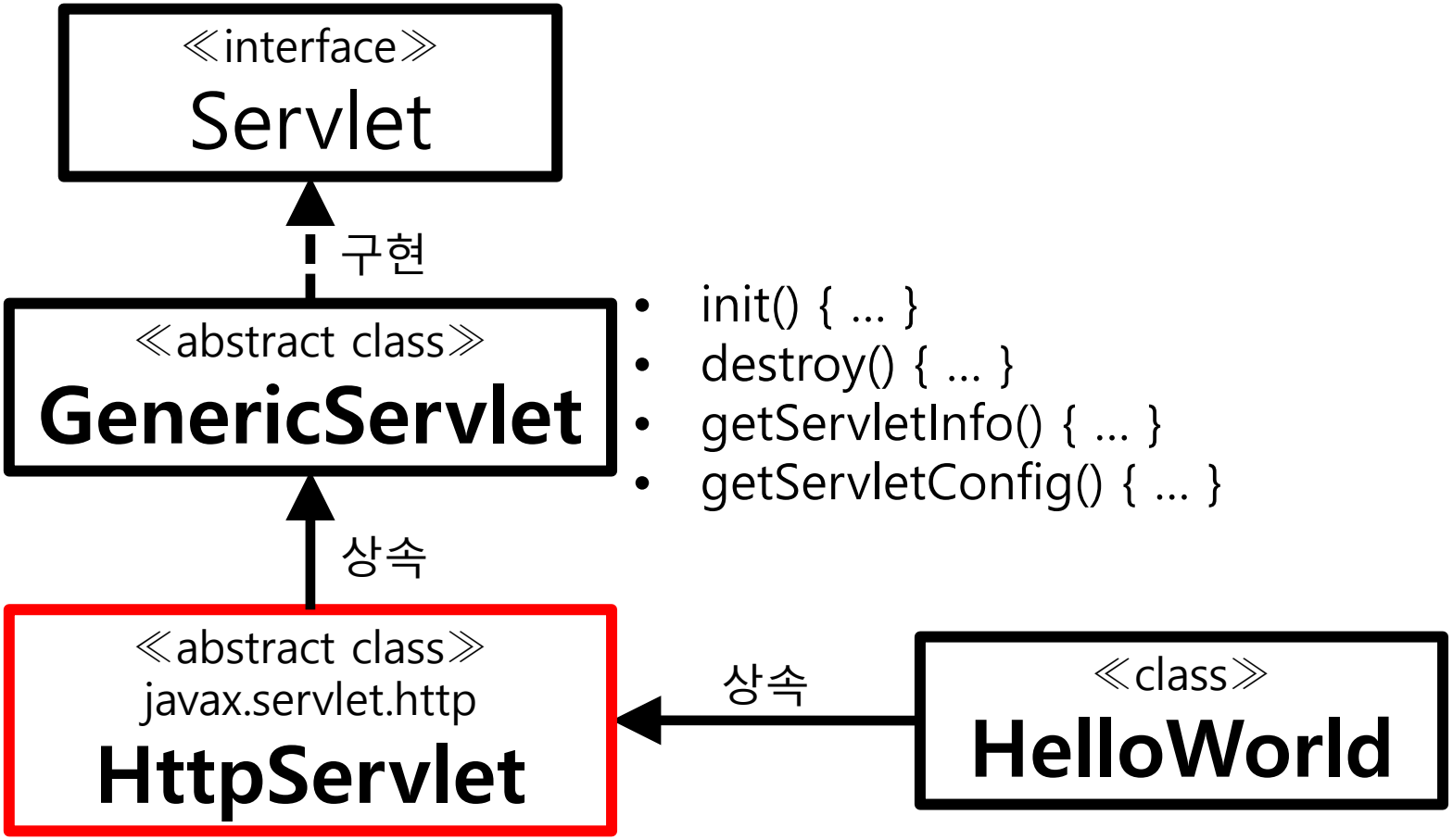
서블릿 만들기


≪interface≫
Servlet
구현
≪abstract class≫
GenericServlet
상속
≪abstract class≫
javax.servlet.http
HttpServlet
상속
≪class≫
HelloWorld

서블릿 만들기

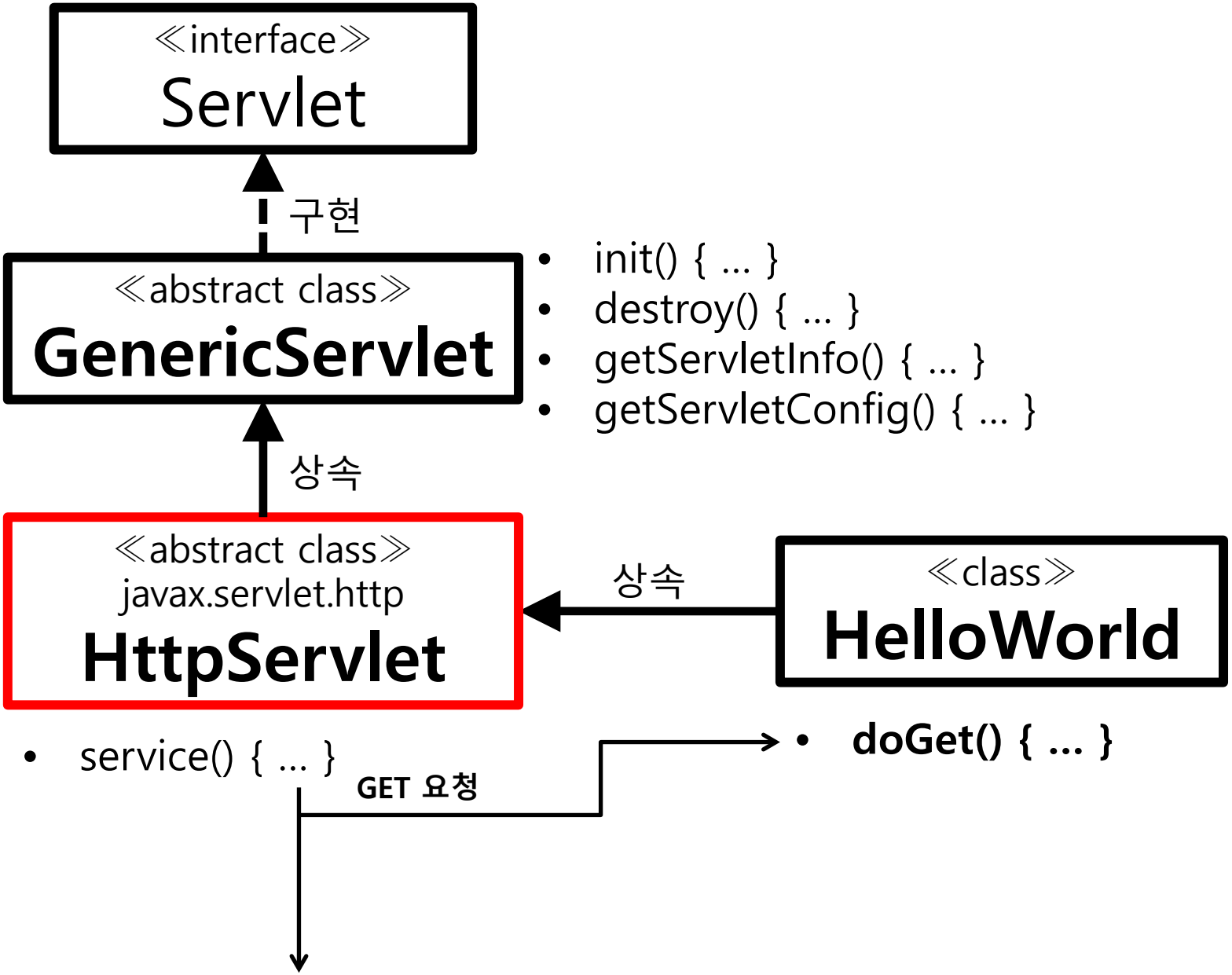

≪interface≫
Servlet
구현
≪abstract class≫
GenericServlet
init() { … }
destroy() { … }
getServletInfo() { … }
getServletConfig() { … }
상속
≪abstract class≫
javax.servlet.http
HttpServlet
상속
≪class≫
HelloWorld

서블릿 만들기

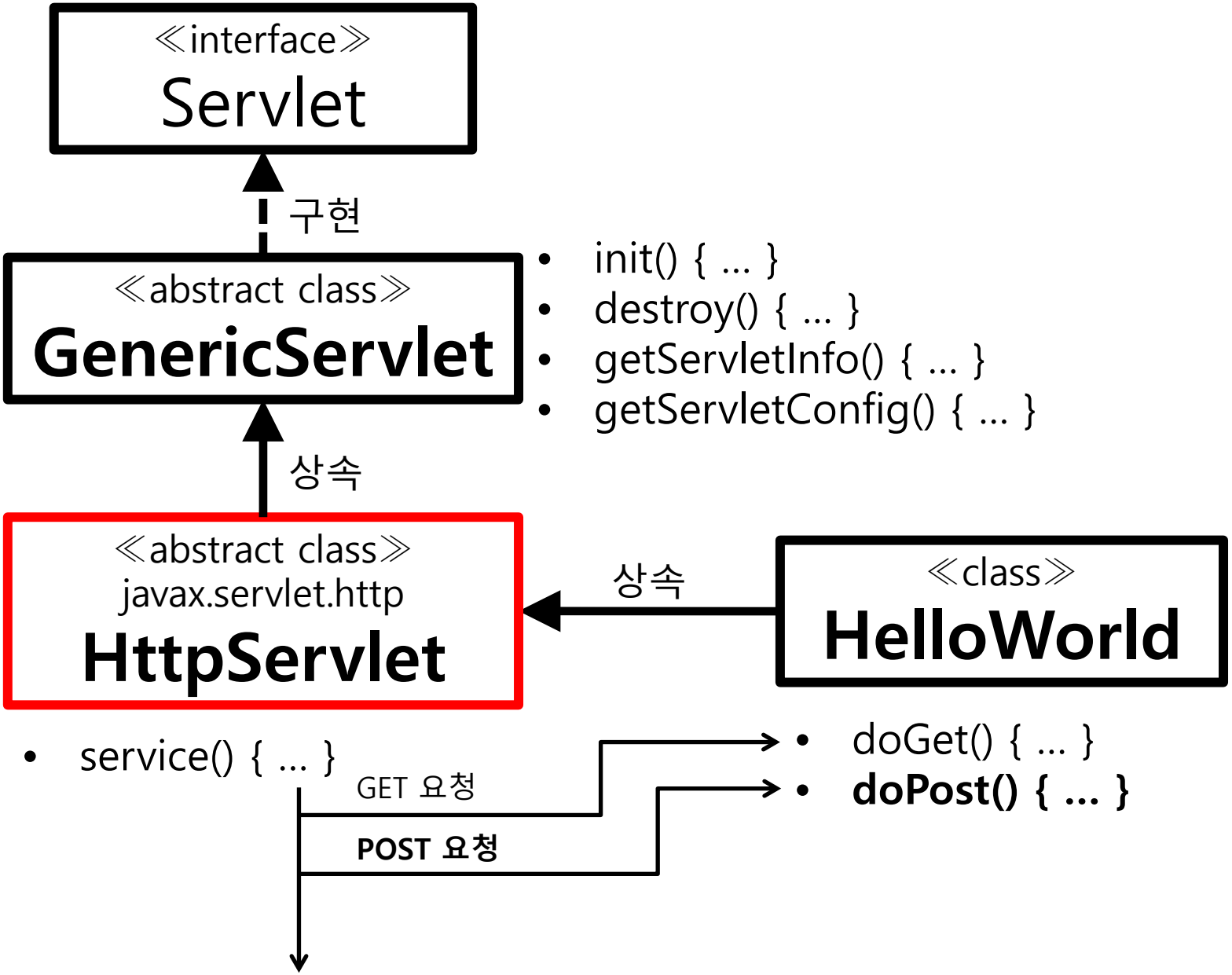


- service() { … }

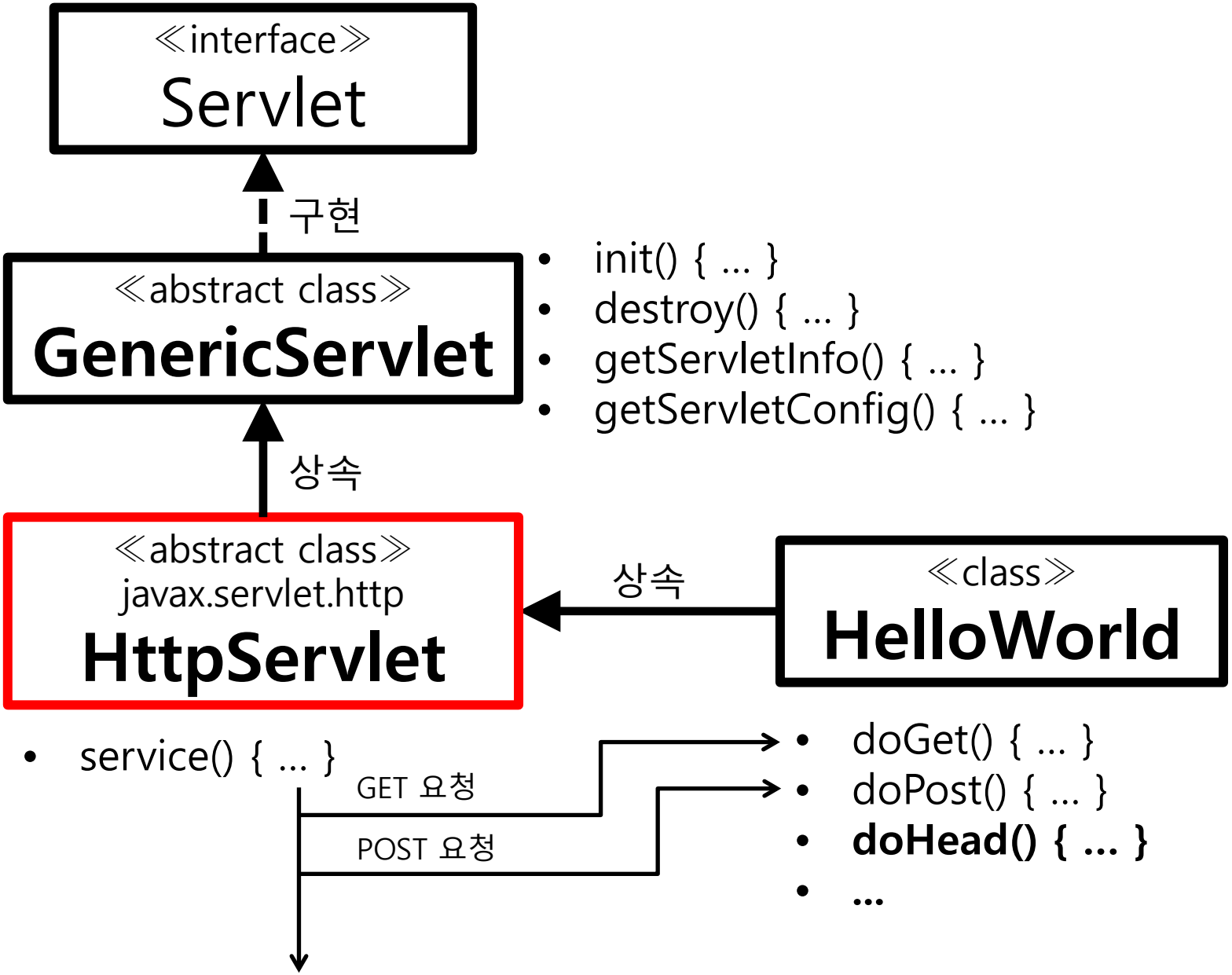
서블릿 만들기


≪interface≫
Servlet
구현
≪abstract class≫
GenericServlet
• init() { … }
• destroy() { … }
• getServletInfo() { … }
• getServletConfig() { … }
상속
≪abstract class≫
javax.servlet.http
HttpServlet
상속
≪class≫
HelloWorld
• service() { … }
GET 요청
• doGet() { … }

서블릿 만들기


≪interface≫
Servlet
구현
≪abstract class≫
GenericServlet
• init() { … }
• destroy() { … }
• getServletInfo() { … }
• getServletConfig() { … }
상속
≪abstract class≫
javax.servlet.http
HttpServlet
상속
≪class≫
HelloWorld
• service() { … }
GET 요청
POST 요청
• doGet() { … }
• doPost() { … }

서블릿 만들기


≪interface≫
Servlet
구현
≪abstract class≫
GenericServlet
• init() { … }
• destroy() { … }
• getServletInfo() { … }
• getServletConfig() { … }
상속
≪abstract class≫
javax.servlet.http
HttpServlet
상속
≪class≫
HelloWorld
• service() { … }
GET 요청
POST 요청
• doGet() { … }
• doPost() { … }
• doHead() { … }
• …

한글 입력 값이 깨지는 이유?

회원 등록

이름: 오호라

이메일: ohora@test.com

암호: ••••

추가 취소

회원 등록

이름: 오호라

이메일: ohora@test.com

암호: ••••

추가 취소

추가 버튼
클릭!

HTTP
요청 프로토콜


```
POST /web04/member/add HTTP/1.1
Host: localhost:9999
Connection: keep-alive
Content-Length: 69
Cache-Control: max-age=0
Accept: text/html,application/xhtml+xml, …
…

name=%EC%98%A4%ED%98%B8%EB%9D%BC&email=ohora%40test.com&password=1111
```

POST /web04/member/add HTTP/1.1
Host: localhost:9999
Connection: keep-alive
Content-Length: 69
Cache-Control: max-age=0
Accept: text/html,application/xhtml+xml, …
…

name=%EC%98%A4%ED%98%B8%EB%9D%BC&email=ohora%40test.com&password=1111

입력폼
파라미터 값

```
POST /web04/member/add HTTP/1.1
Host: localhost:9999
Connection: keep-alive
Content-Length: 69
Cache-Control: max-age=0
Accept: text/html,application/xhtml+xml, …
…

name=%EC%98%A4%ED%98%B8%EB%9D%BC&email=ohora%40test.com&password=1111
```


```
POST /web04/member/add HTTP/1.1
Host: localhost:9999
Connection: keep-alive
Content-Length: 69
Cache-Control: max-age=0
Accept: text/html,application/xhtml+xml, …
…

name=%EC%98%A4%ED%98%B8%EB%9D%BC&email=ohora%40test.com&password=1111
```

POST /web04/member/add HTTP/1.1
Host: localhost:9999
Connection: keep-alive
Content-Length: 69
Cache-Control: max-age=0
Accept: text/html,application/xhtml+xml, …
…

name=**%EC%98%A4%ED%98%B8%EB%9D%BC**&email=ohora%40test.com&password=1111

URL 인코딩


```
POST /web04/member/add HTTP/1.1
Host: localhost:9999
Connection: keep-alive
Content-Length: 69
Cache-Control: max-age=0
Accept: text/html,application/xhtml+xml, …
…

name=%EC%98%A4%ED%98%B8%EB%9D%BC&email=ohora%40test.com&password=1111
```

UTF-8 ➔ URL 인코딩

name=**%EC%98%A4%ED%98%B8%EB%9D%BC**&email=ohora%40test.com&password=1111

≪서블릿≫

HelloWorld

name=**%EC%98%A4%ED%98%B8%EB%9D%BC**&email=ohora%40test.com&password=1111

URL
디코딩➔
UTF-8

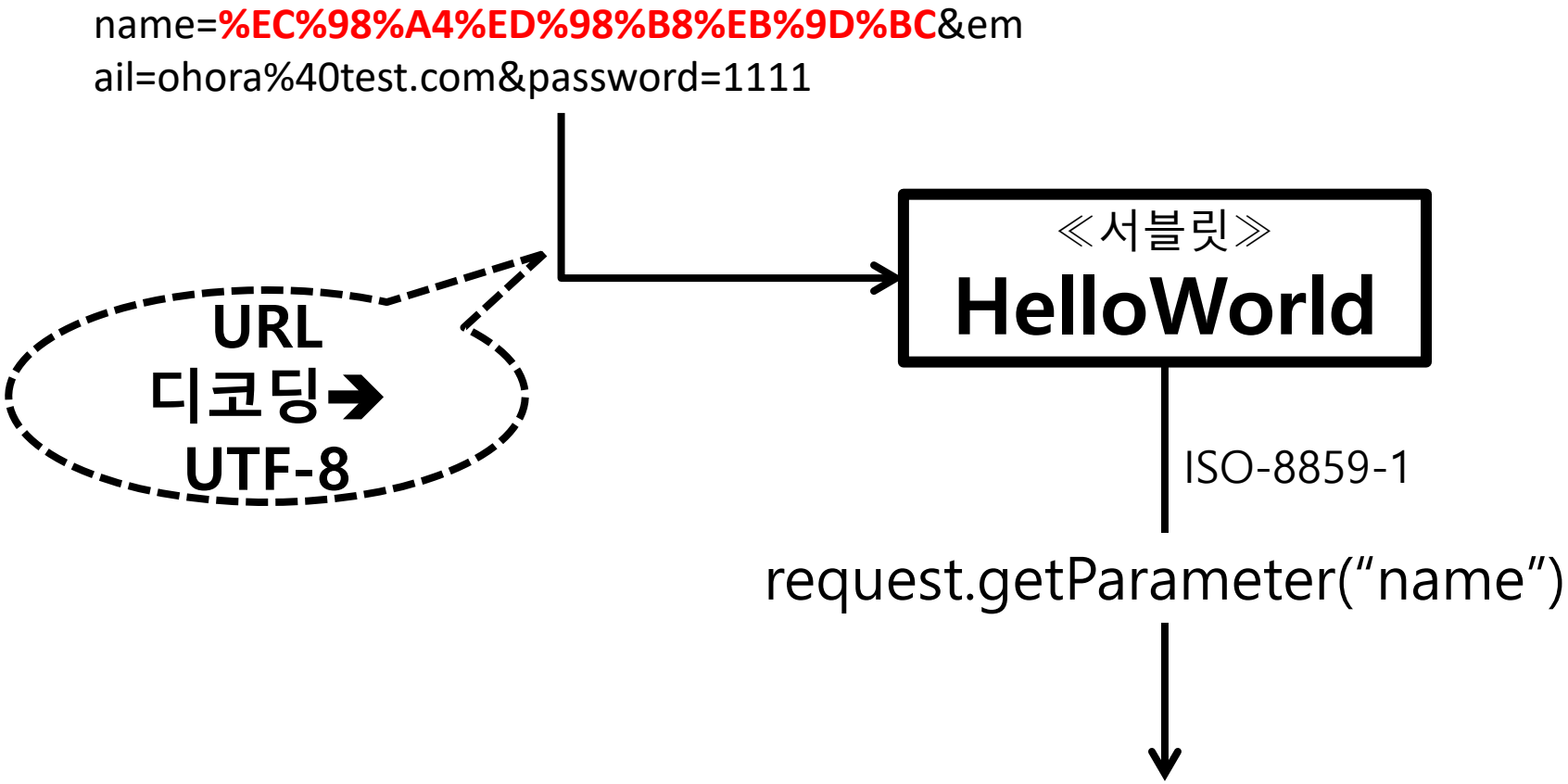
≪서블릿≫
HelloWorld

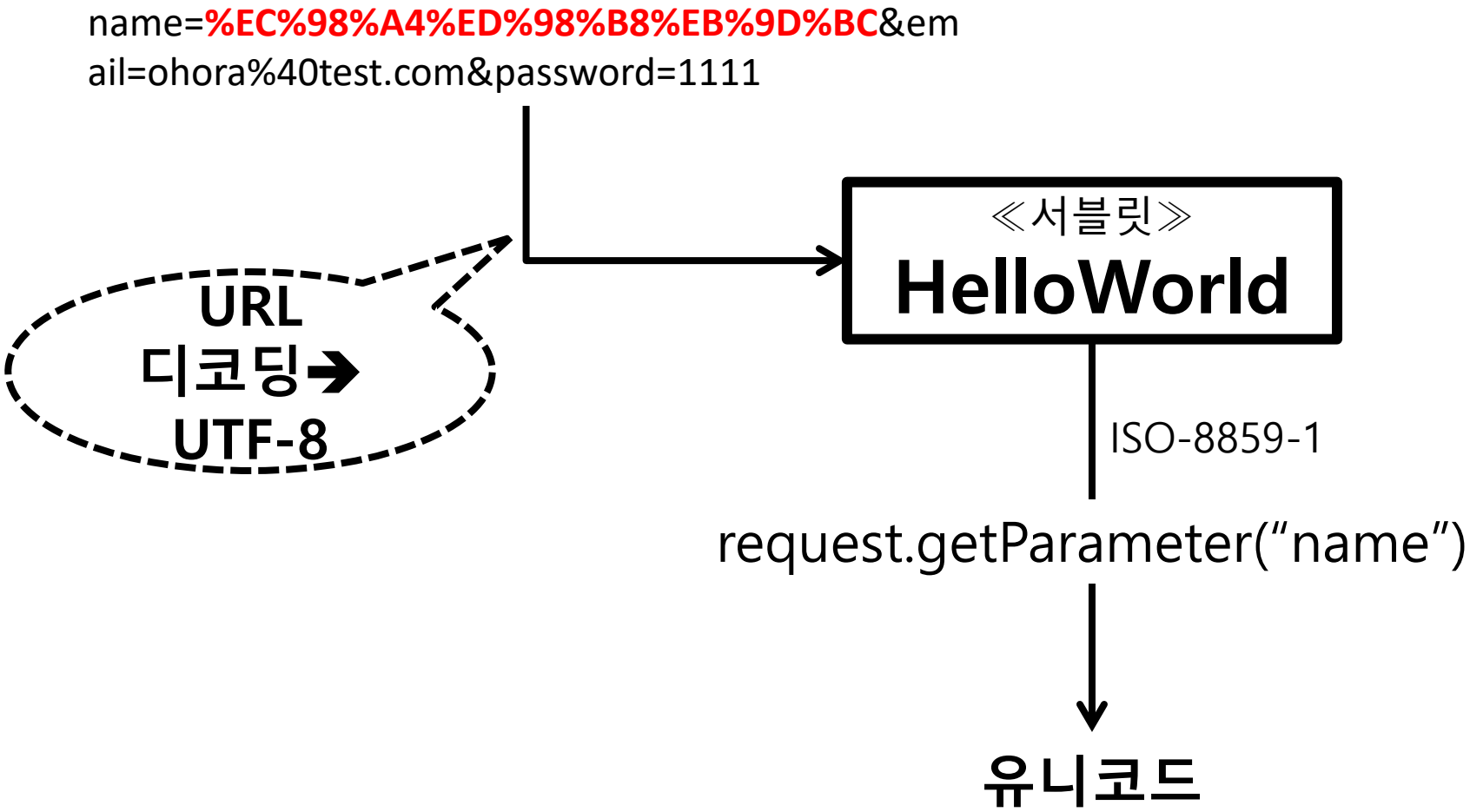
name=**%EC%98%A4%ED%98%B8%EB%9D%BC**&email=ohora%40test.com&password=1111

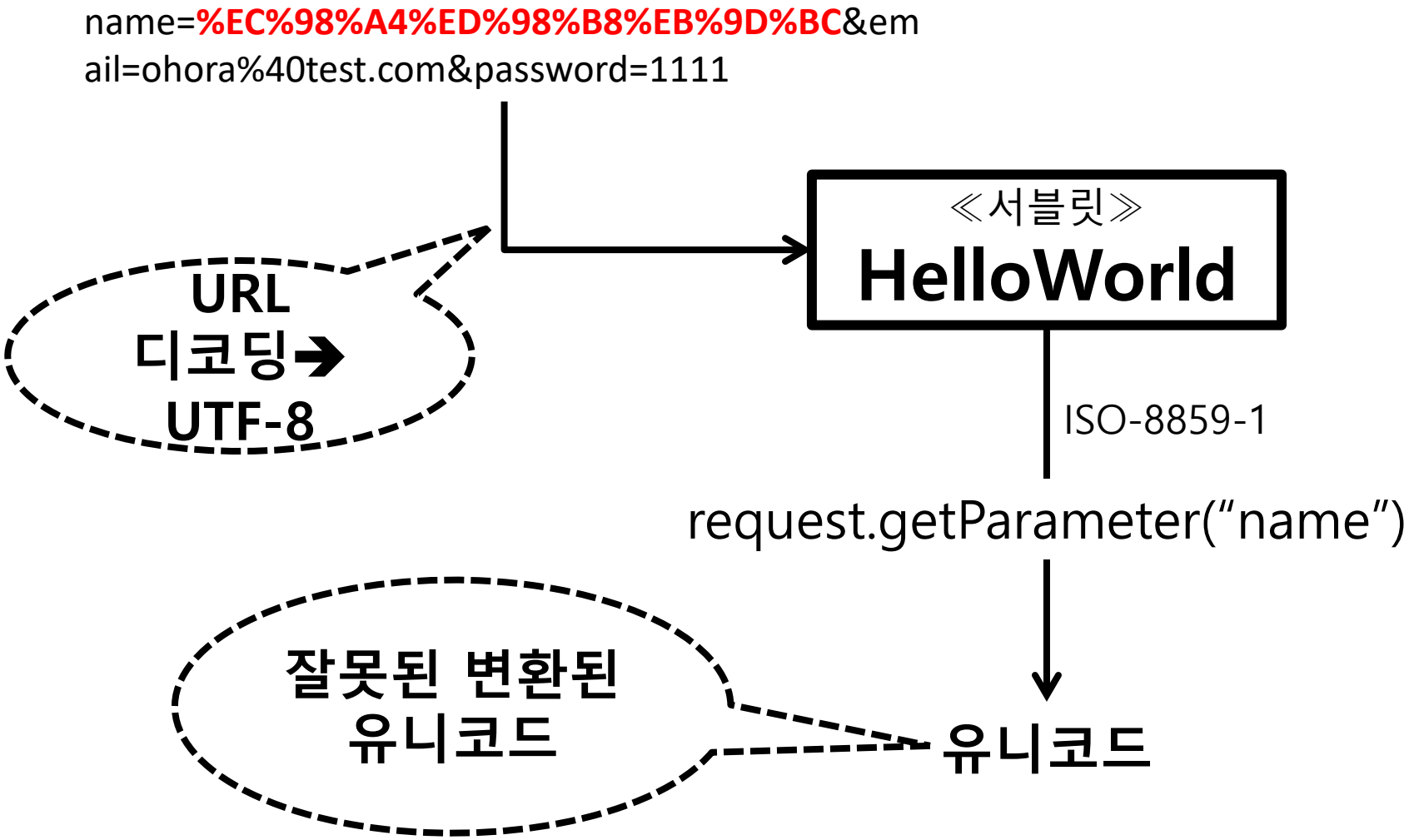
URL
디코딩➔
UTF-8

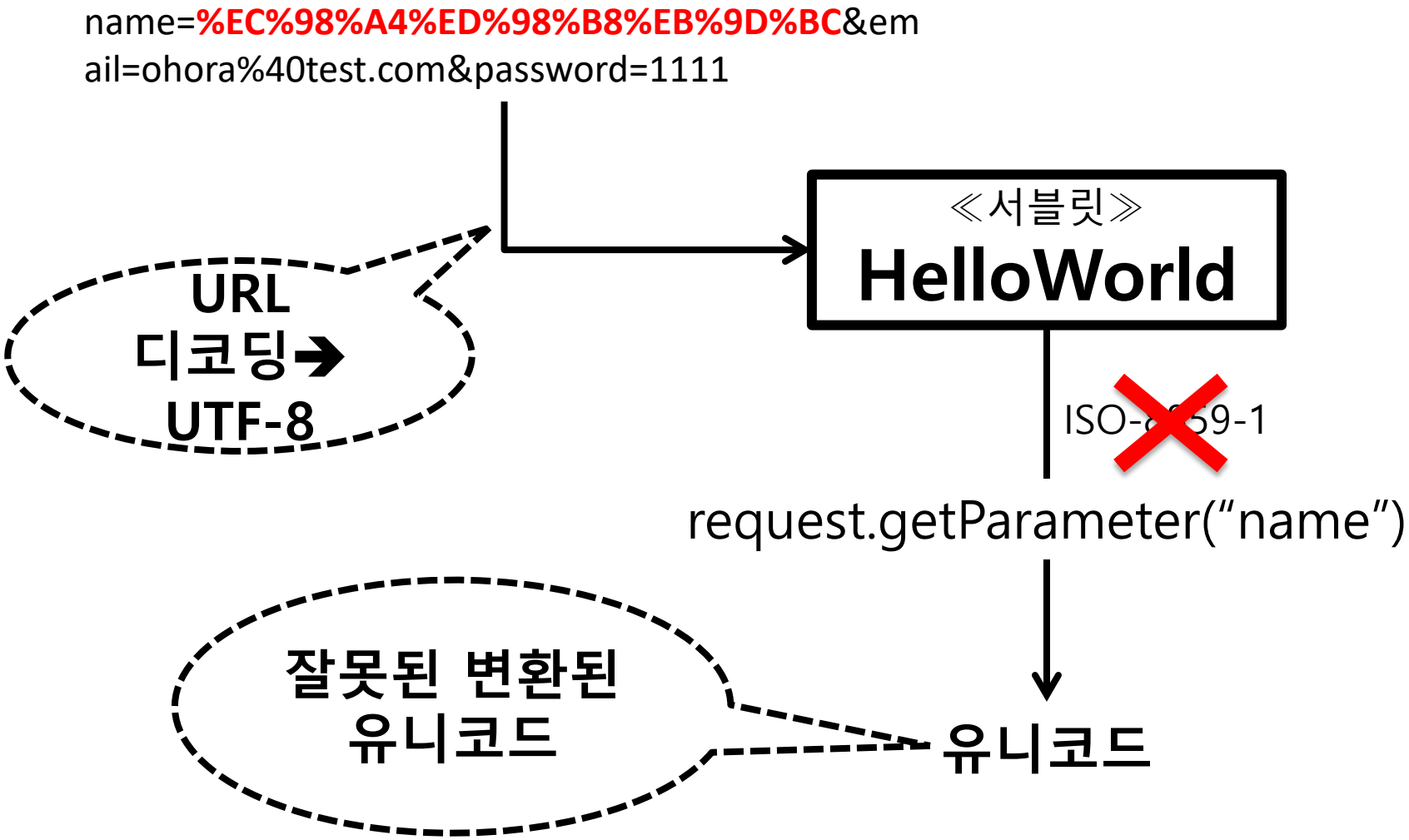
≪서블릿≫
HelloWorld

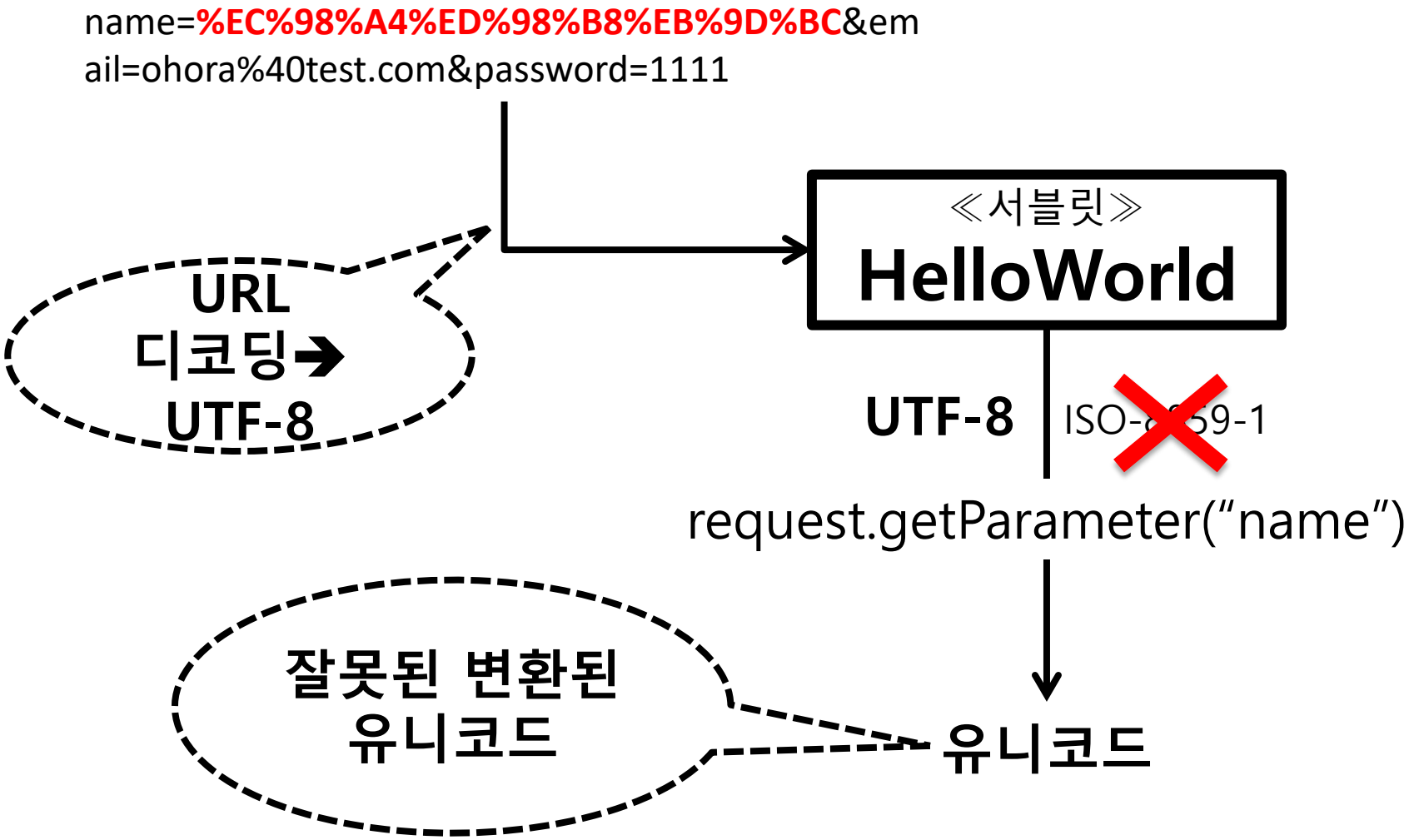
request.getParameter("name")


name=%EC%98%A4%ED%98%B8%EB%9D%BC&email=ohora%40test.com&password=1111
URL 디코딩➔ UTF-8
≪서블릿≫
HelloWorld
ISO-8859-1
request.getParameter("name")


name=%EC%98%A4%ED%98%B8%EB%9D%BC&email=ohora%40test.com&password=1111
URL 디코딩➔ UTF-8
≪서블릿≫
HelloWorld
ISO-8859-1
request.getParameter("name")
유니코드


name=%EC%98%A4%ED%98%B8%EB%9D%BC&email=ohora%40test.com&password=1111
URL 디코딩➔ UTF-8
≪서블릿≫
HelloWorld
ISO-8859-1
request.getParameter("name")
잘못된 변환된 유니코드
유니코드


name=%EC%98%A4%ED%98%B8%EB%9D%BC&email=ohora%40test.com&password=1111
URL 디코딩➔ UTF-8
≪서블릿≫
HelloWorld
ISO-8859-1
request.getParameter("name")
잘못된 변환된 유니코드
유니코드


name=%EC%98%A4%ED%98%B8%EB%9D%BC&email=ohora%40test.com&password=1111
URL 디코딩➔ UTF-8
≪서블릿≫
HelloWorld
UTF-8
ISO-8859-1
request.getParameter("name")
잘못된 변환된 유니코드
유니코드

name=**%EC%98%A4%ED%98%B8%EB%9D%BC**&email=ohora%40test.com&password=1111

URL
디코딩➔
UTF-8

≪서블릿≫
HelloWorld

UTF-8 | ~~ISO-8859-1~~

request.getParameter("name")

↓

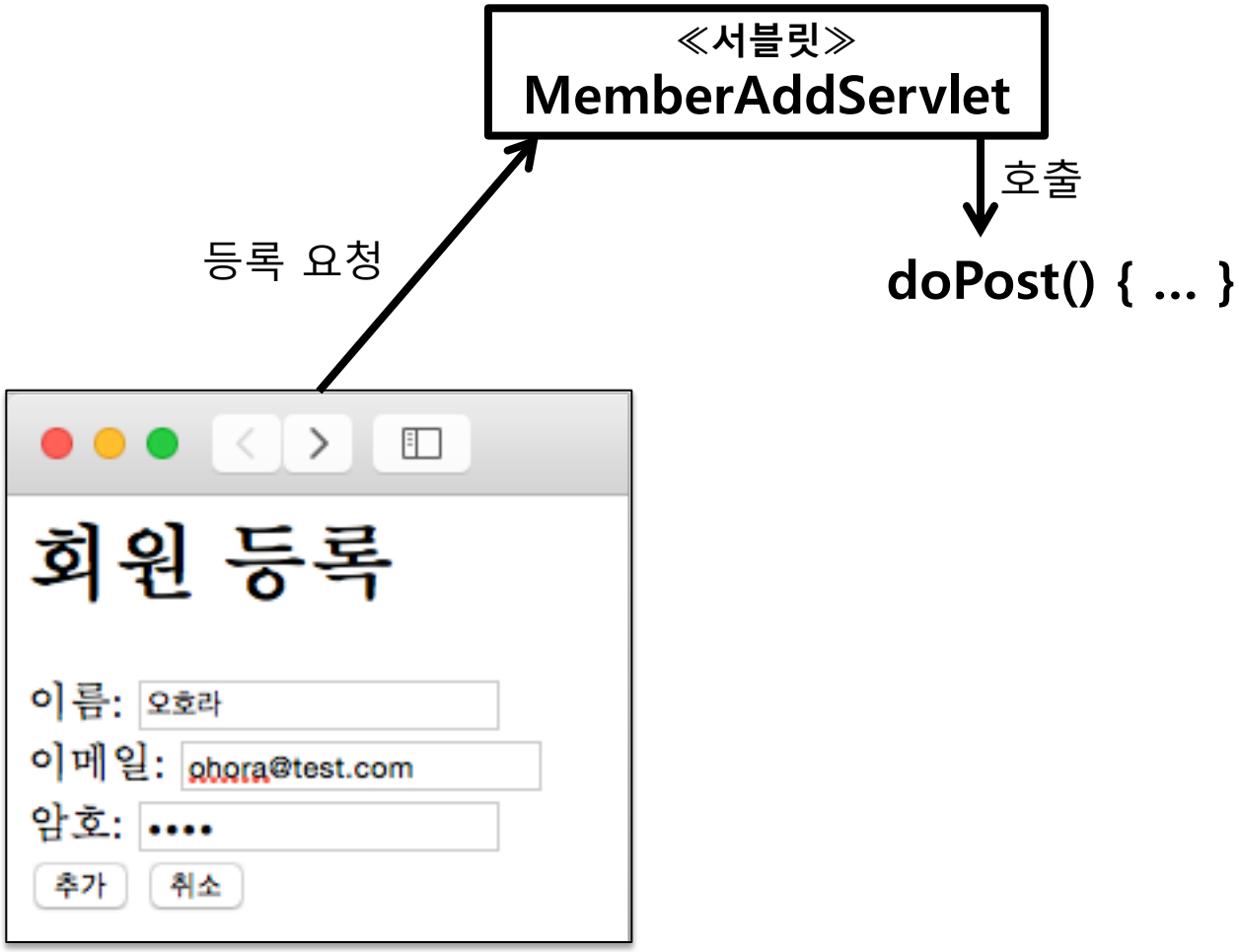
올바르게 변환된
유니코드

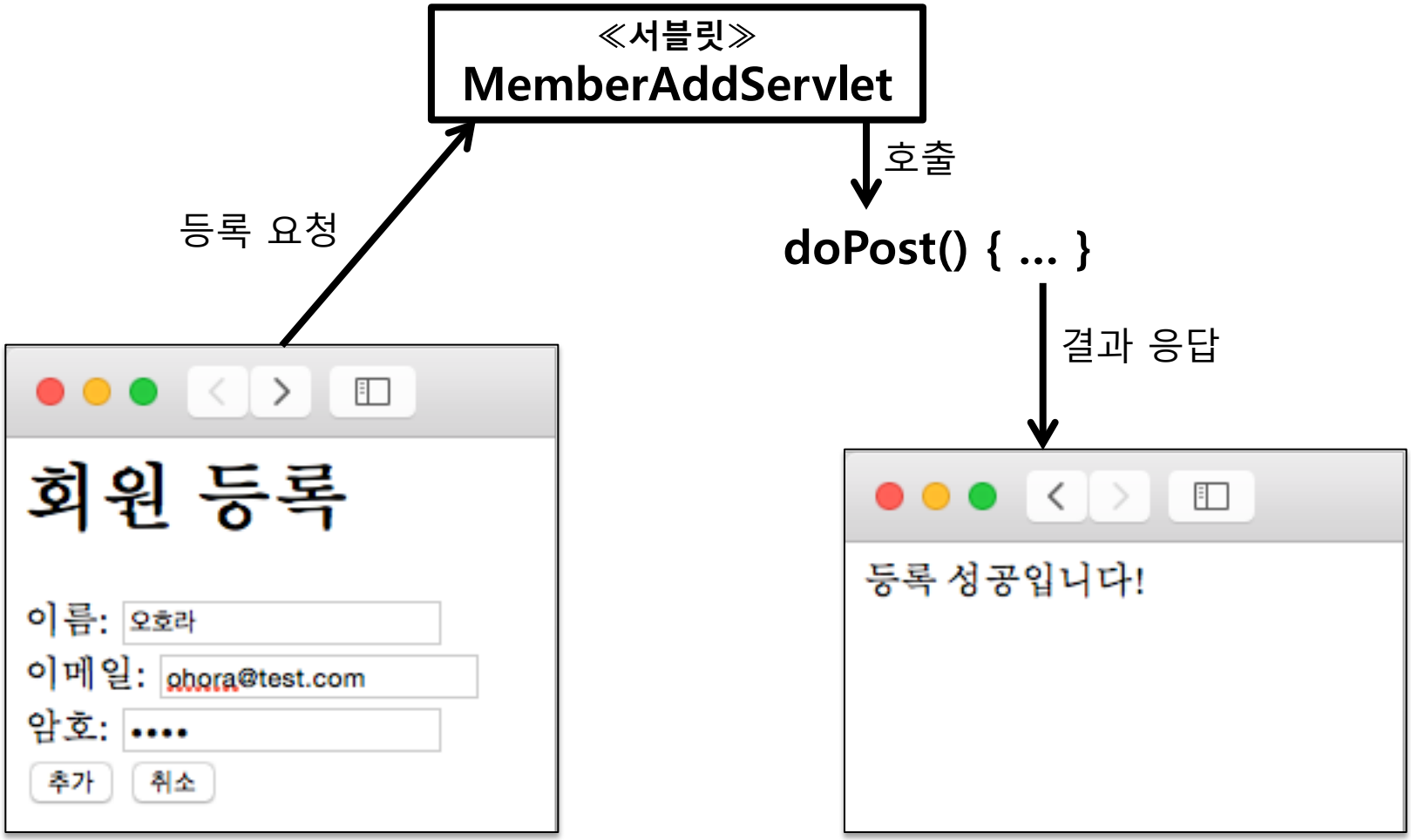
4.5 리프래시


회원 등록
이름:
오호라
이메일:
ohora@test.com
암호:
추가
취소

≪서블릿≫
MemberAddServlet

등록 요청


≪서블릿≫
MemberAddServlet
호출
doPost() { ... }
등록 요청
회원 등록
이름:
오호라
이메일:
ohora@test.com
암호:
추가
취소


≪서블릿≫
MemberAddServlet
호출
등록 요청
doPost() { ... }
결과 응답
회원 등록
이름:
오호라
이메일:
ohora@test.com
암호:
추가
취소
등록 성공입니다!

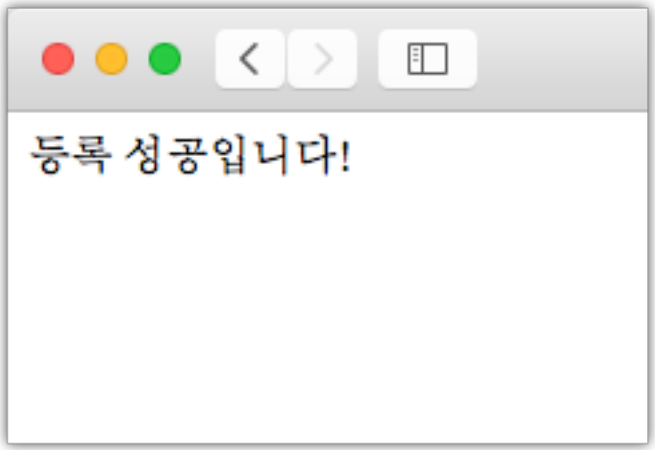
등록한 다음은?

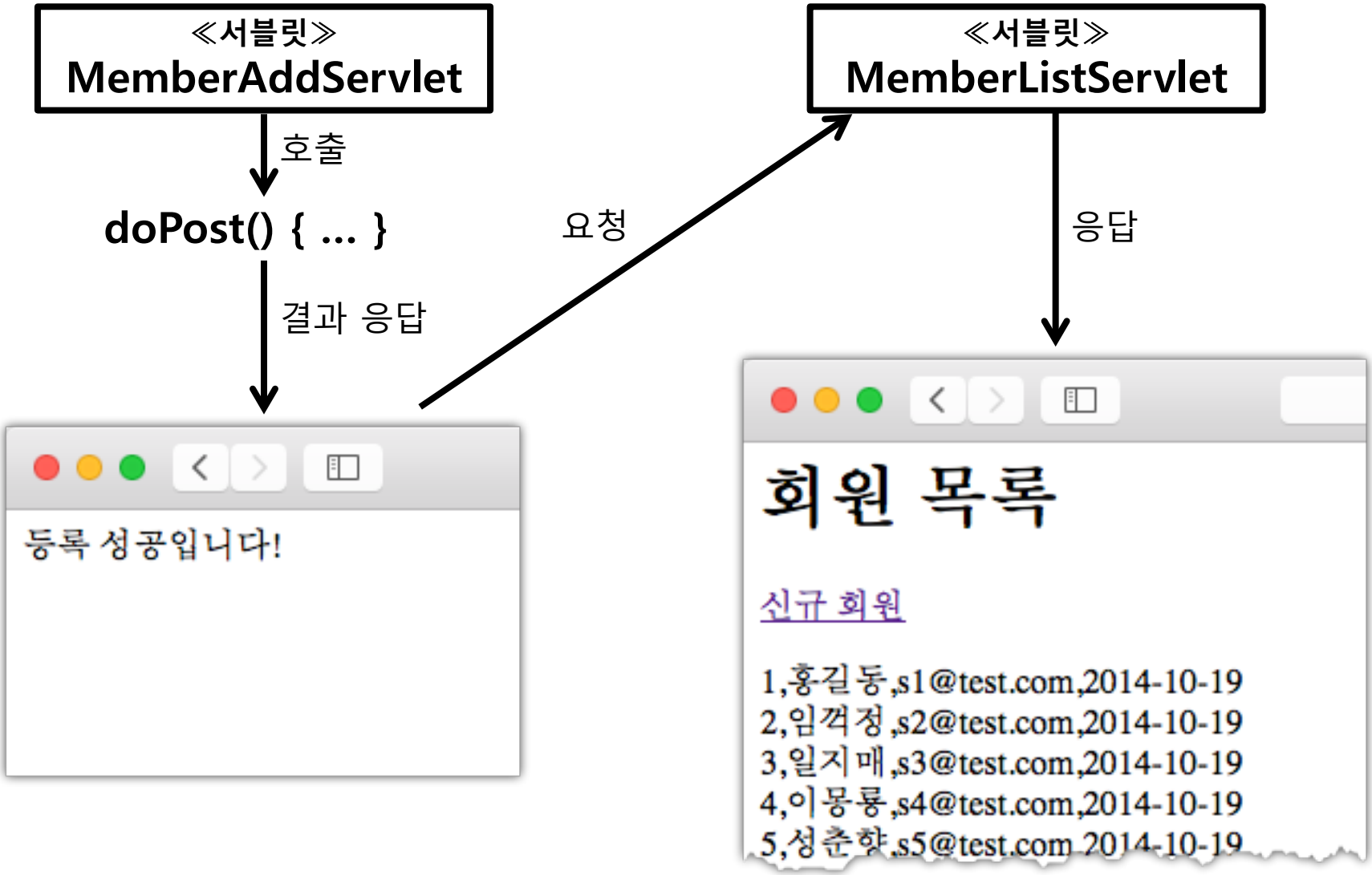
≪서블릿≫
MemberAddServlet

호출

doPost() { ... }

결과 응답




≪서블릿≫
MemberAddServlet
호출
doPost() { ... }
결과 응답
요청
≪서블릿≫
MemberListServlet
응답
등록 성공입니다!
회원 목록
신규 회원
1,홍길동,s1@test.com,2014-10-19
2,임꺽정,s2@test.com,2014-10-19
3,일지매,s3@test.com,2014-10-19
4,이몽룡,s4@test.com,2014-10-19
5,성춘향,s5@test.com,2014-10-19

자동으로 목록화면으로 이동하려면?

리프래시

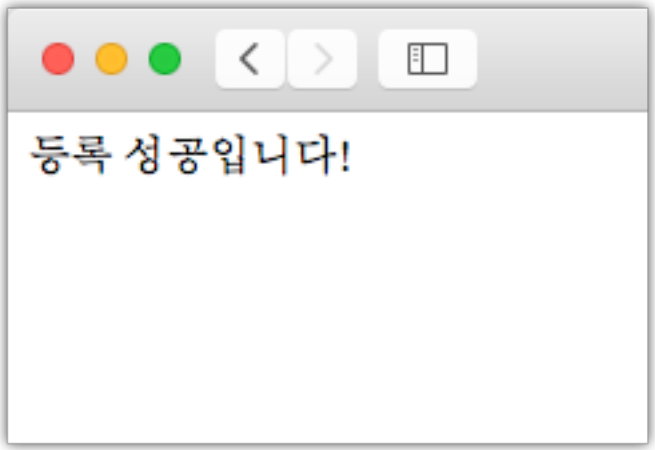
≪서블릿≫
MemberAddServlet

호출

doPost() { ... }

결과 응답

HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Refresh: 1;url=list
Content-Type: text/html;charset=UTF-8
...

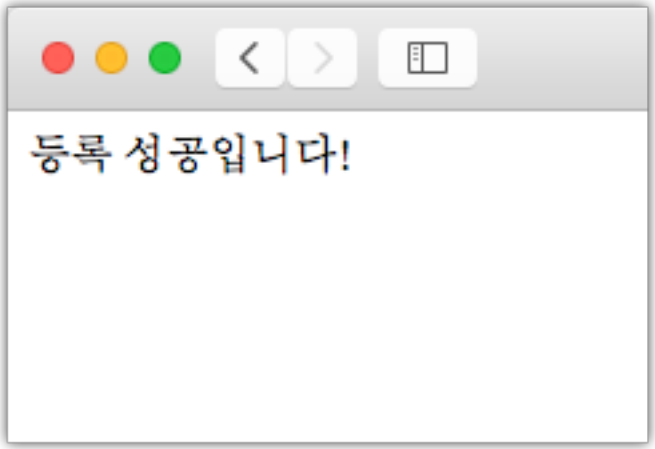


≪서블릿≫
MemberAddServlet

호출

doPost() { ... }

결과 응답



HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Refresh: 1;url=list
Content-Type: text/html;charset=UTF-8
...

Refresh 헤더 추가

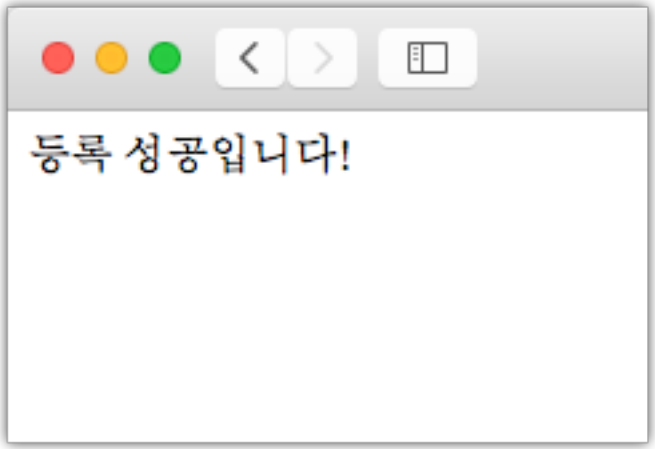
≪서블릿≫
MemberAddServlet

호출

doPost() { ... }

결과 응답

HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Refresh: 1;url=list
Content-Type: text/html;charset=UTF-8
...



Refresh: 1;url=list

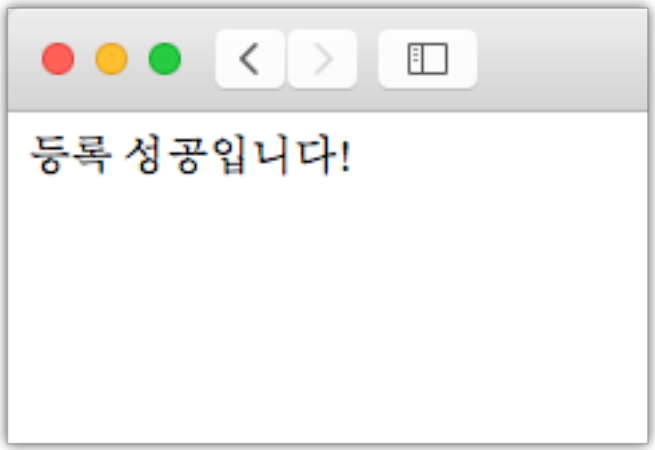
≪서블릿≫
MemberAddServlet

호출

doPost() { ... }

결과 응답

HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Refresh: 1;url=list
Content-Type: text/html;charset=UTF-8
...



Refresh: 1;url=list

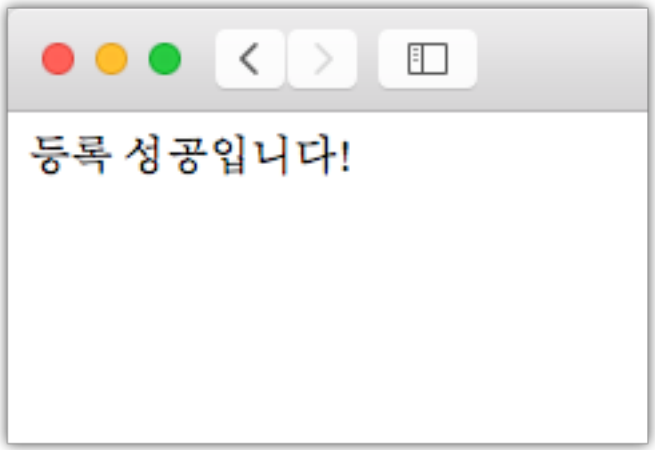
헤더 이름

≪서블릿≫
MemberAddServlet

호출

doPost() { … }

결과 응답



HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Refresh: 1;url=list
Content-Type: text/html; charset=UTF-8
…

경과
시간(초)

Refresh: 1;url=list

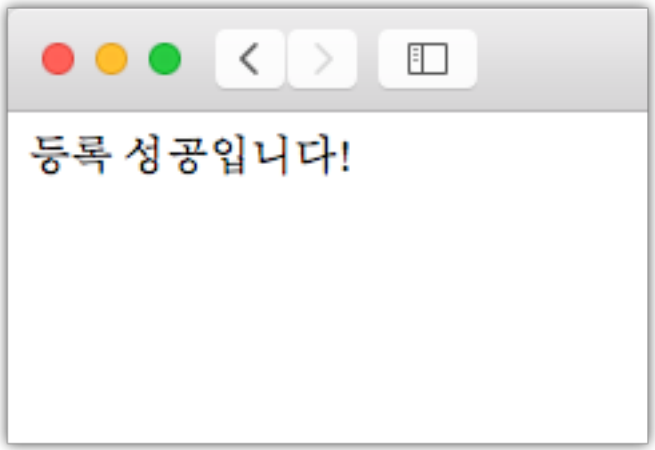
≪서블릿≫
MemberAddServlet

호출

doPost() { ... }

결과 응답

HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Refresh: 1;url=list
Content-Type: text/html;charset=UTF-8
...



Refresh: 1;url=list

다시 요청할
URL 주소(상대 경로)

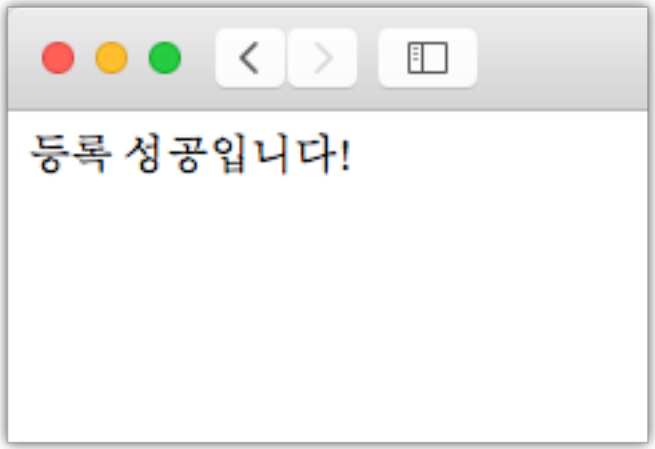
≪서블릿≫
MemberAddServlet

호출

doPost() { … }

결과 응답

HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Refresh: 1;url=list
Content-Type: text/html;charset=UTF-8
…



Refresh: 1;url=list

- **현재 경로: /member/add**

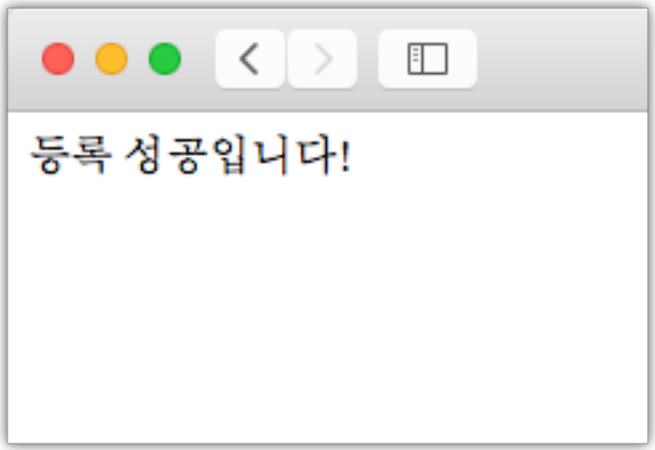
≪서블릿≫
MemberAddServlet

호출

doPost() { ... }

결과 응답

HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Refresh: 1;url=list
Content-Type: text/html;charset=UTF-8
...



Refresh: 1;url=list

- 현재 경로: /member/add
- **다시 요청할 경로: /member/list**

Refresh 응답 헤더를 추가하는 방법?

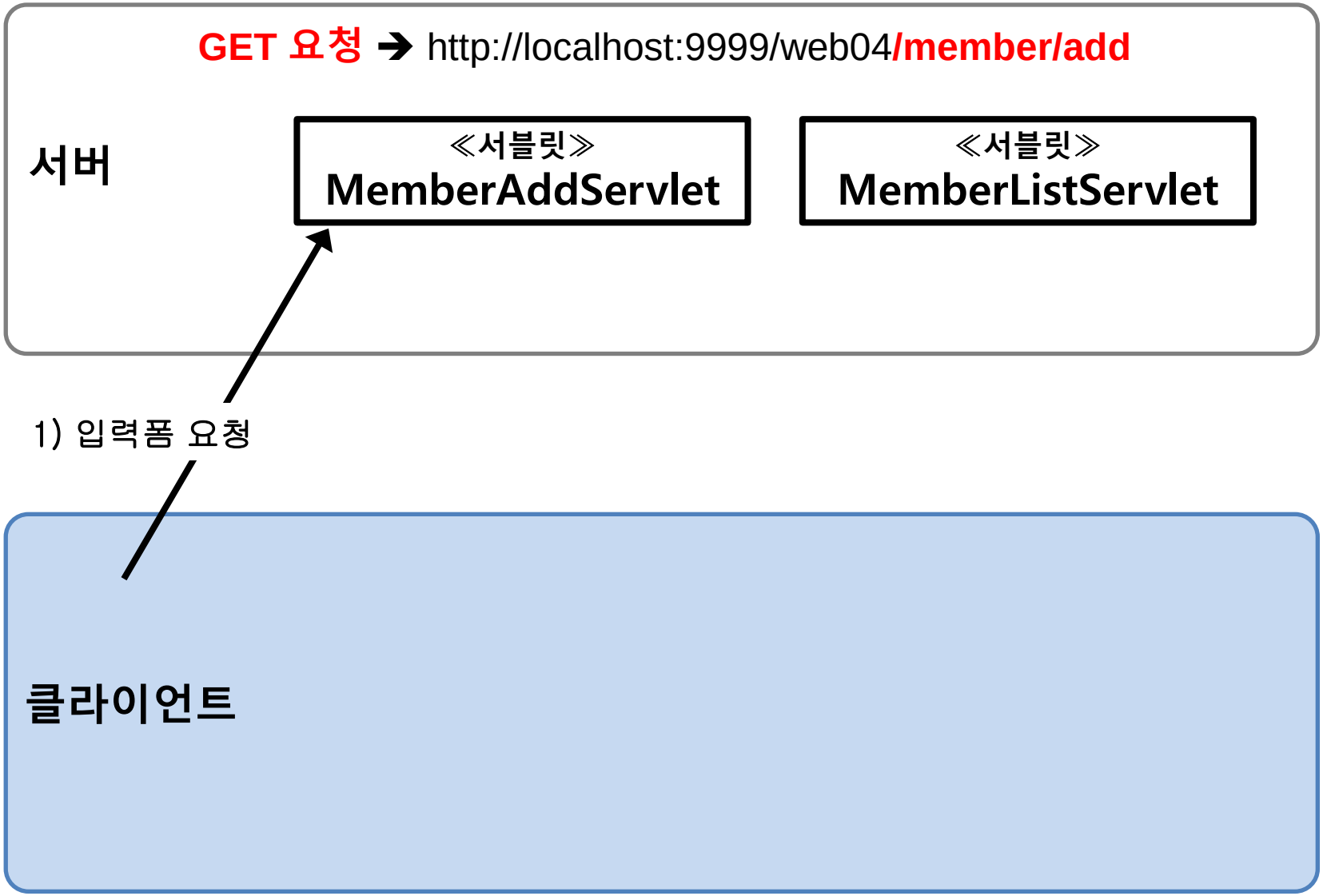
response.**addHeader**(“Refresh”, “1;url=list”);

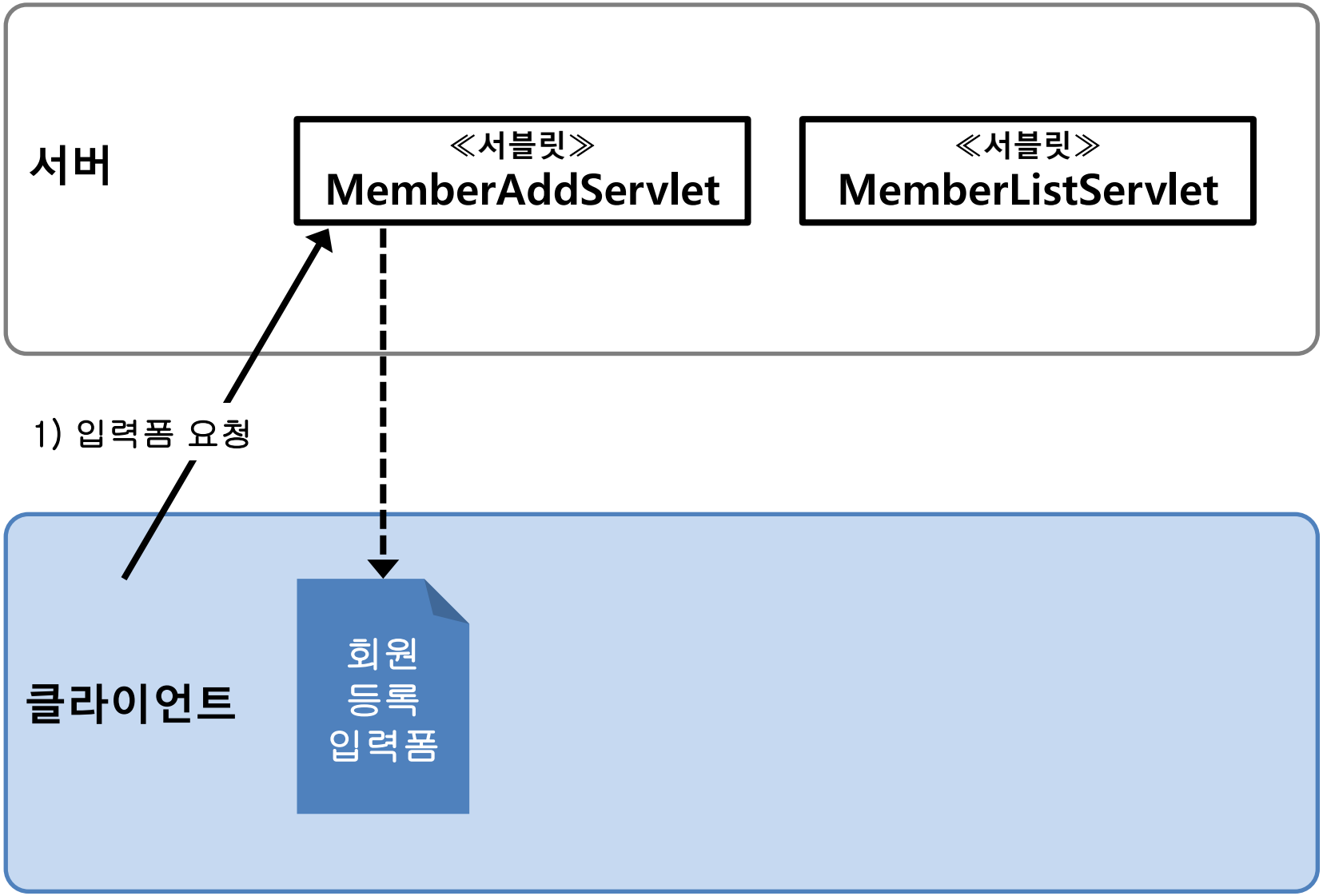
response.**addHeader**(“Refresh”, “1;url=list”);

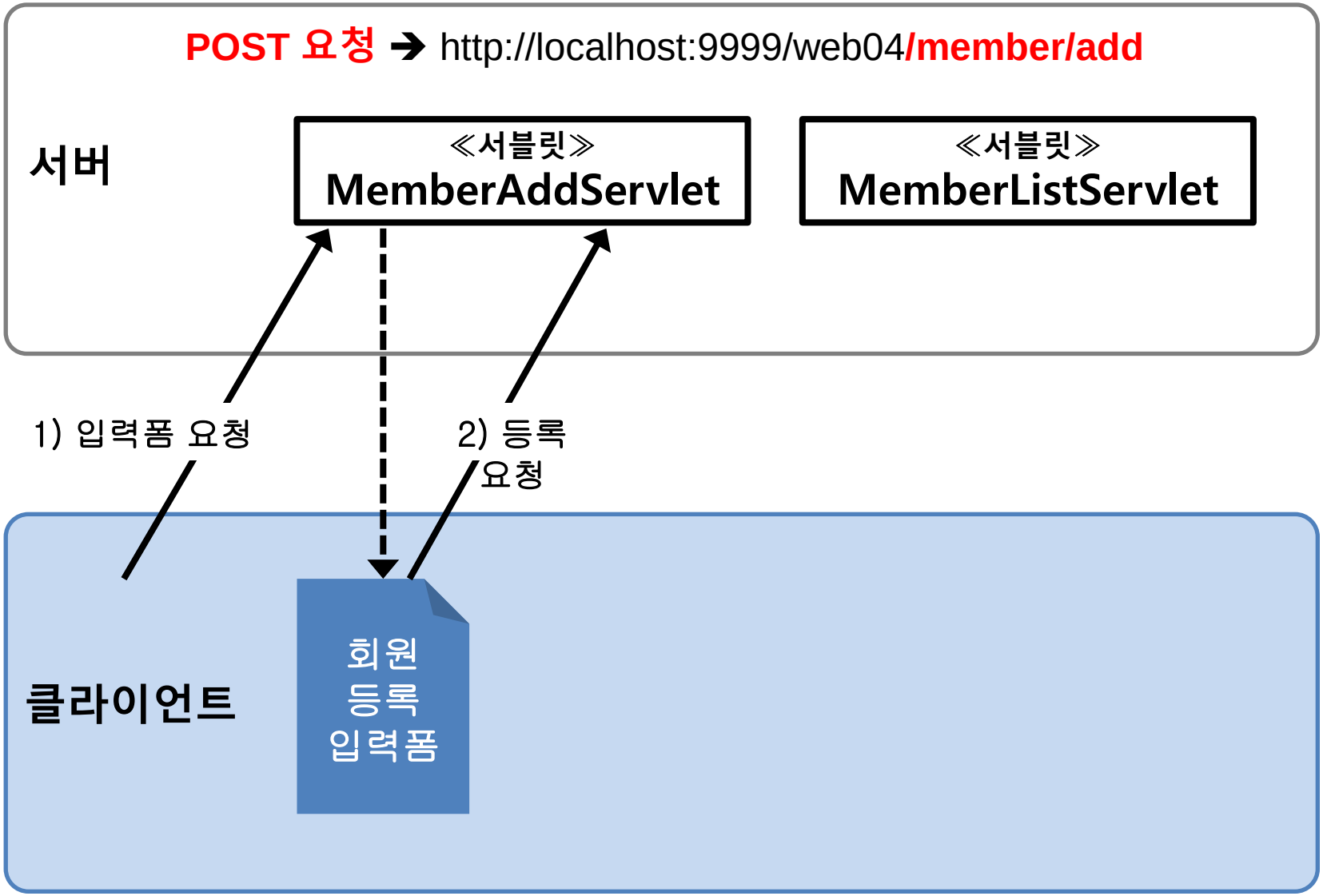
or

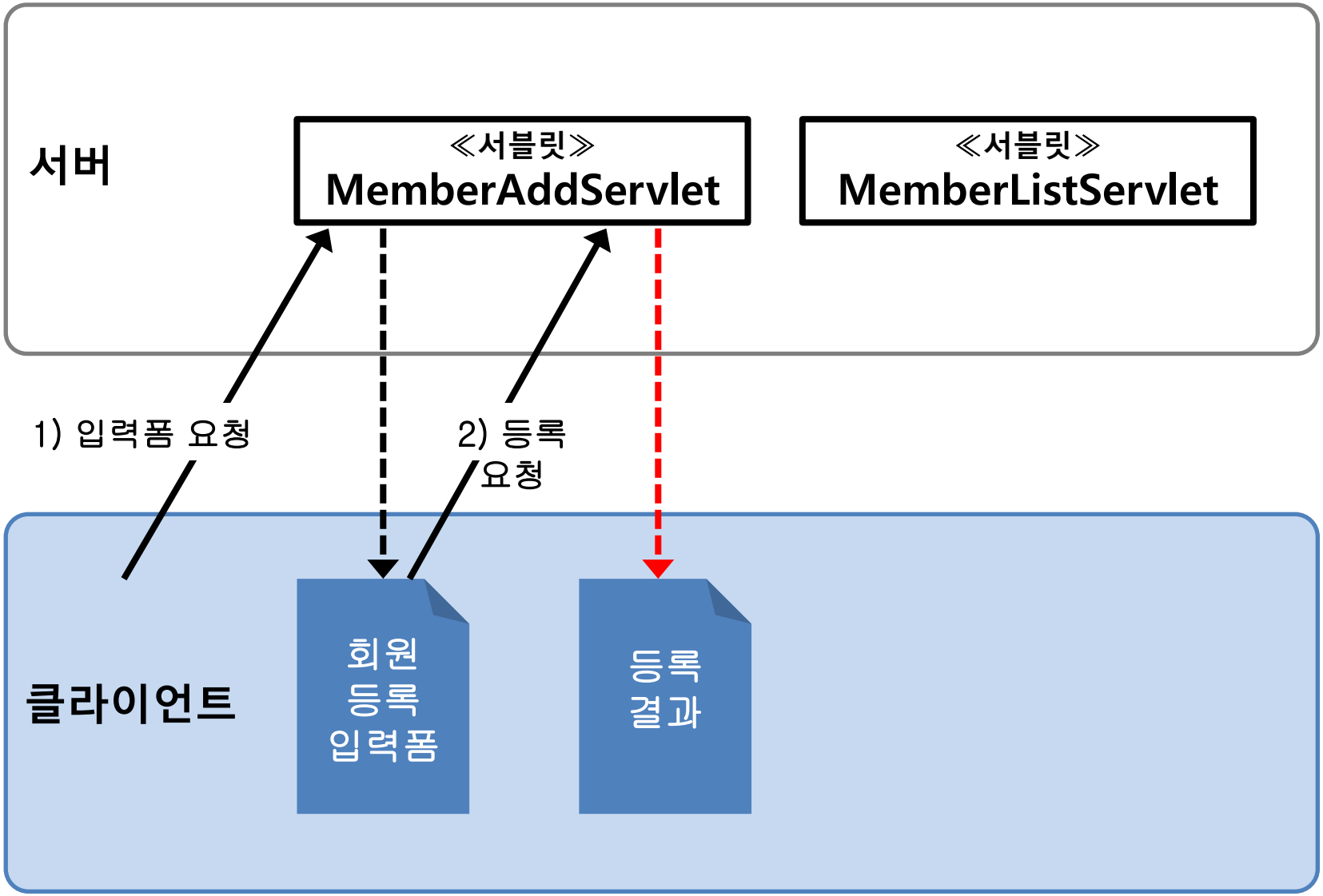
response.**setHeader**(“Refresh”, “1;url=list”);

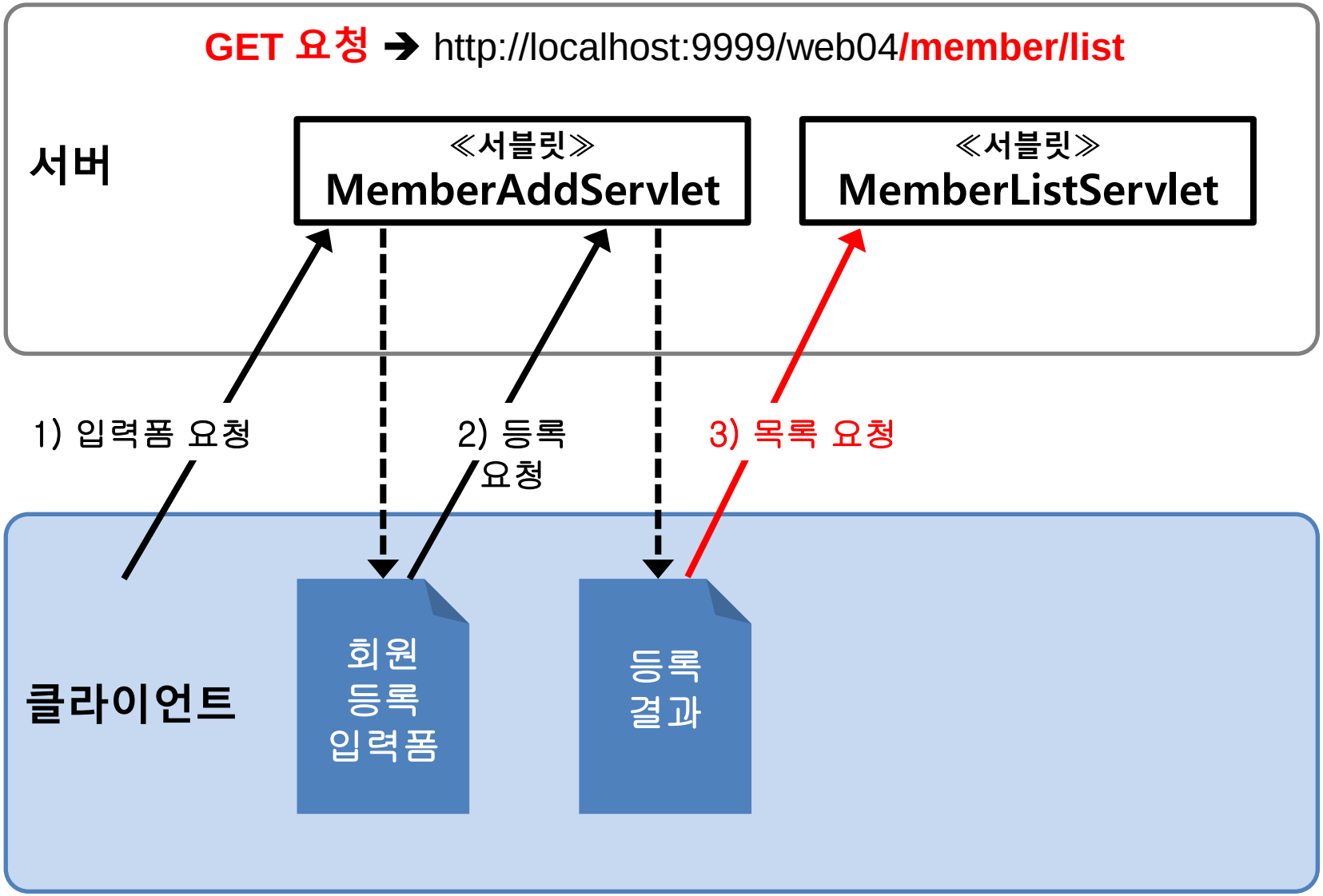
웹브라우저와 웹서버의 요청-응답 흐름

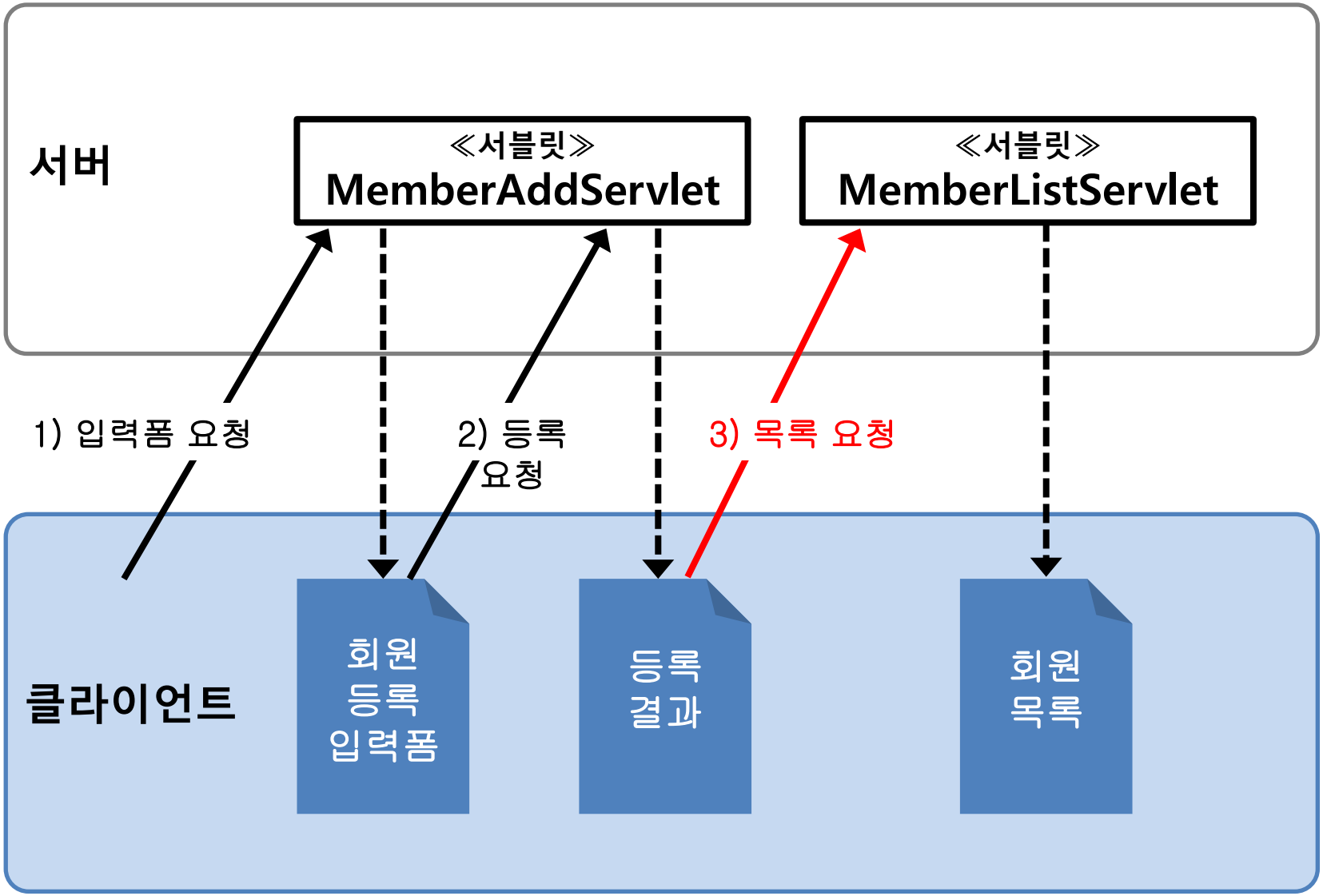

GET 요청 ➔ http://localhost:9999/web04/member/add
서버
≪서블릿≫
MemberAddServlet
≪서블릿≫
MemberListServlet
1) 입력폼 요청
클라이언트


서버
≪서블릿≫
MemberAddServlet
≪서블릿≫
MemberListServlet
1) 입력폼 요청
클라이언트
회원
등록
입력폼


POST 요청 ➔ http://localhost:9999/web04/member/add
서버
≪서블릿≫
MemberAddServlet
≪서블릿≫
MemberListServlet
1) 입력폼 요청
2) 등록 요청
클라이언트
회원 등록 입력폼


서버
≪서블릿≫
MemberAddServlet
≪서블릿≫
MemberListServlet
1) 입력폼 요청
2) 등록
요청
클라이언트
회원
등록
입력폼
등록
결과


GET 요청 ➔ http://localhost:9999/web04/member/list
서버
≪서블릿≫
MemberAddServlet
≪서블릿≫
MemberListServlet
1) 입력폼 요청
2) 등록 요청
3) 목록 요청
클라이언트
회원 등록 입력폼
등록 결과


서버
≪서블릿≫
MemberAddServlet
≪서블릿≫
MemberListServlet
1) 입력폼 요청
2) 등록 요청
3) 목록 요청
클라이언트
회원 등록 입력폼
등록 결과
회원 목록

Refresh를 구현하는 또 다른 방법!

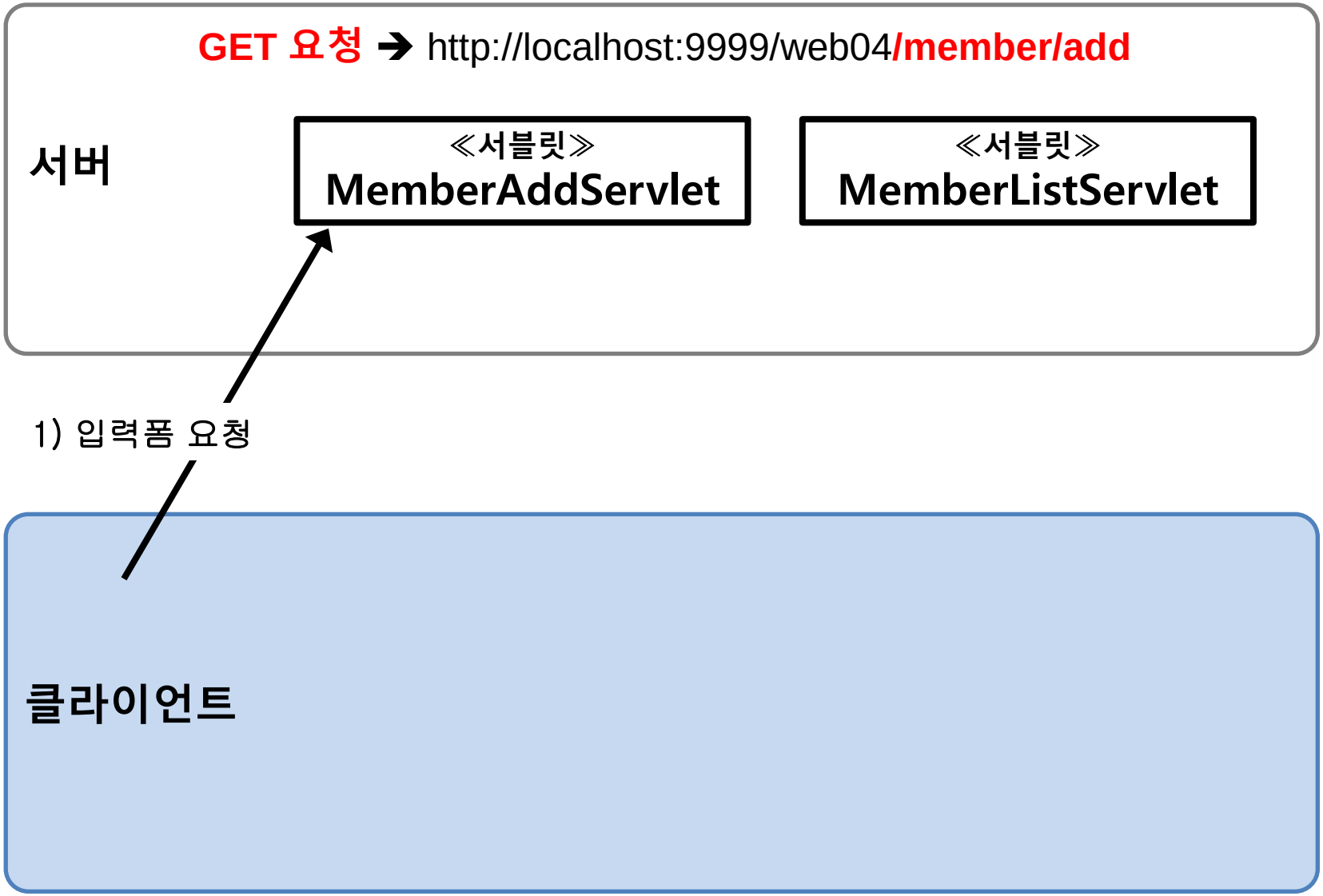
HTML 문에
Refresh 정보 삽입

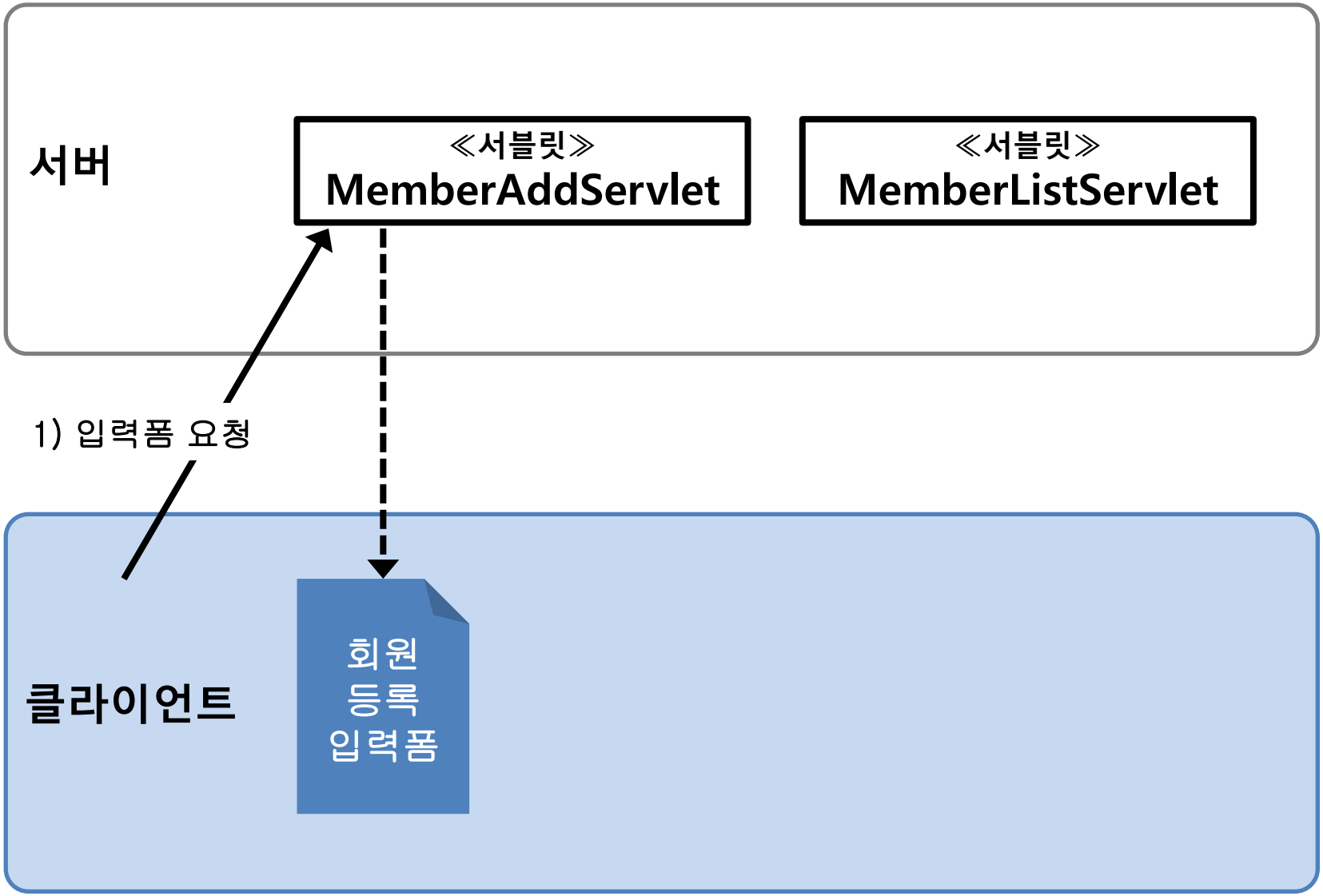
```
<head>
<title>회원등록결과</title>
<meta http-equiv='Refresh' content='1; url=list’>
</head>
…
```

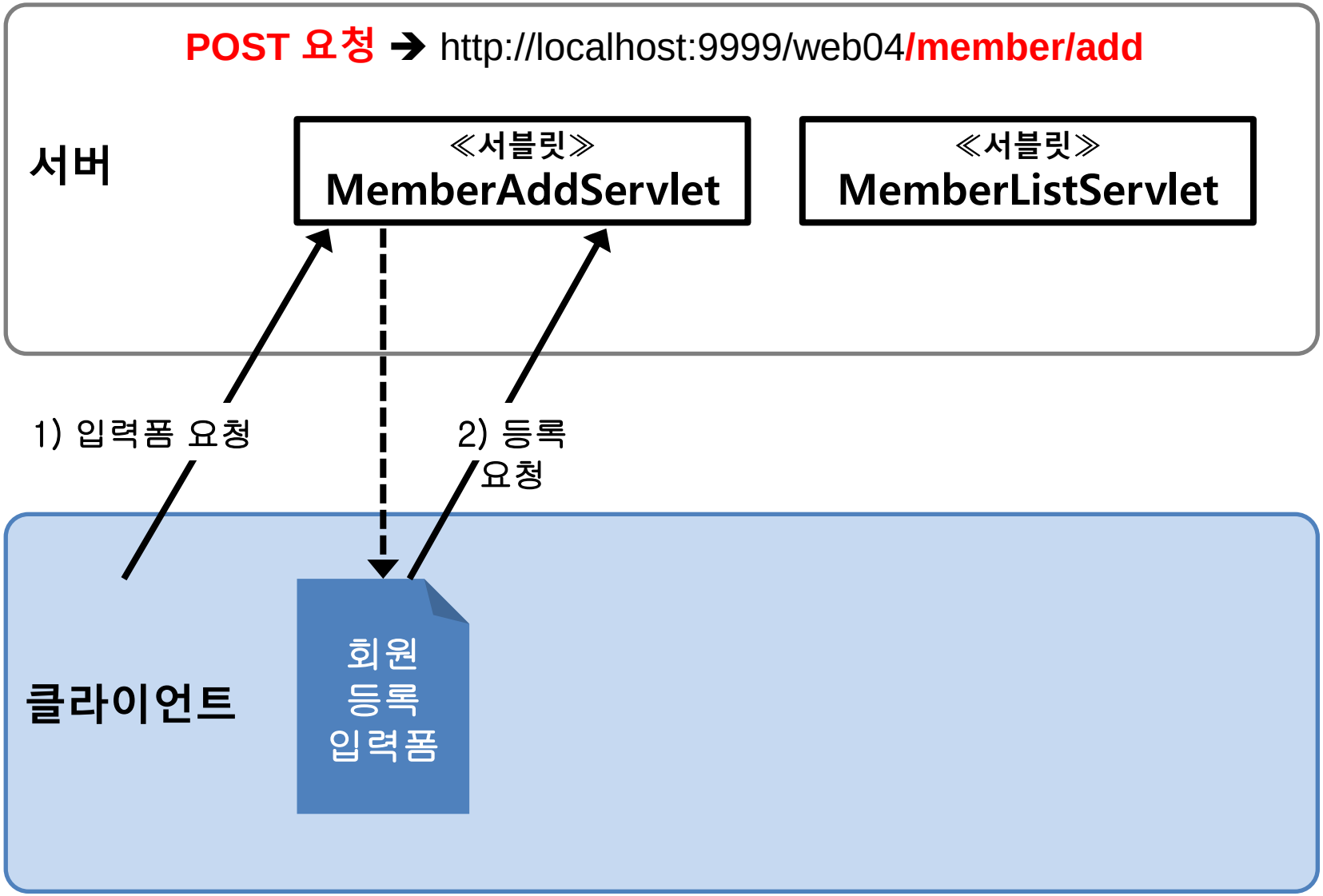
```
<head>
<title>회원등록결과</title>
<meta http-equiv='Refresh' content='1; url=list’>
</head>
…
```

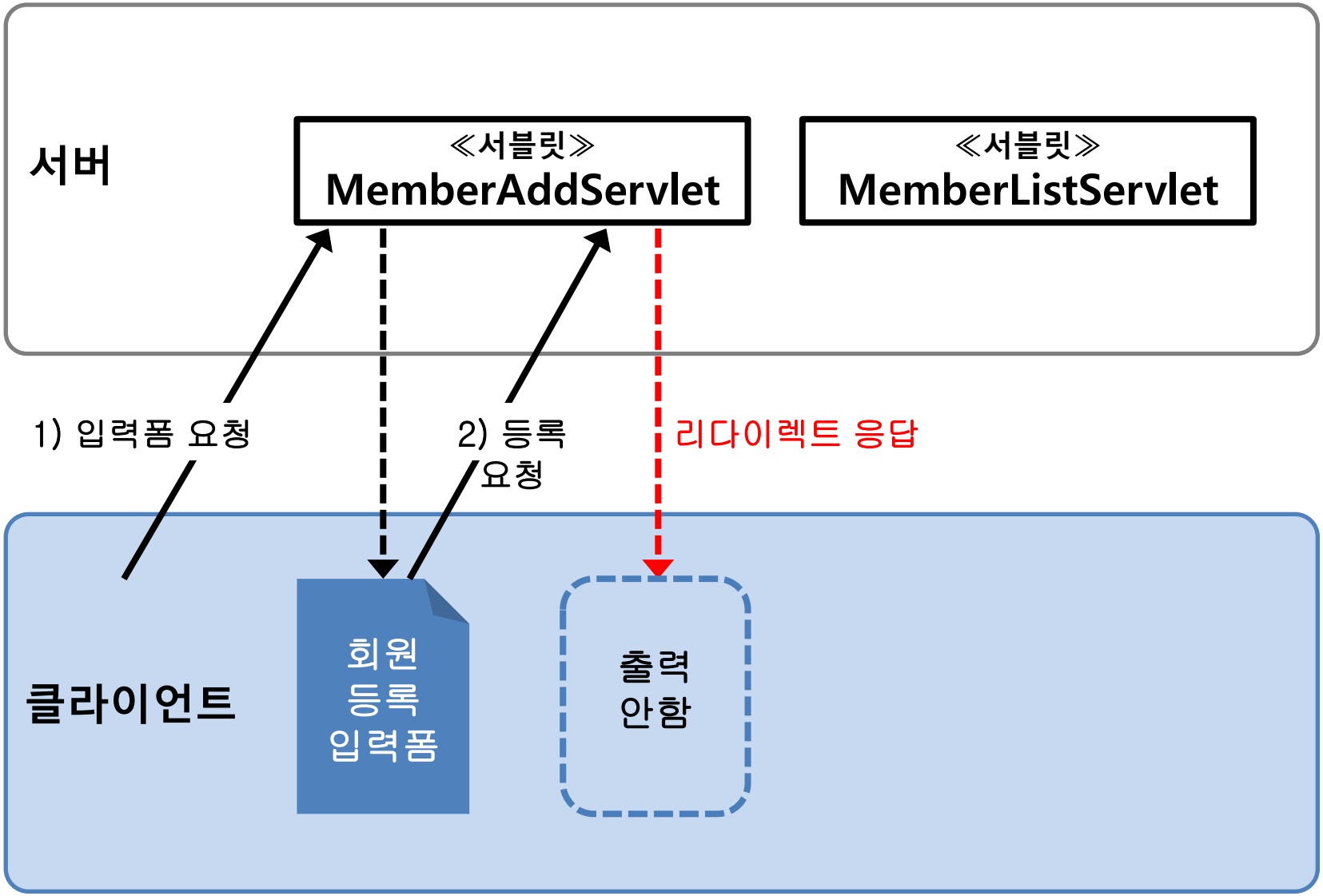
메타 태그 삽입

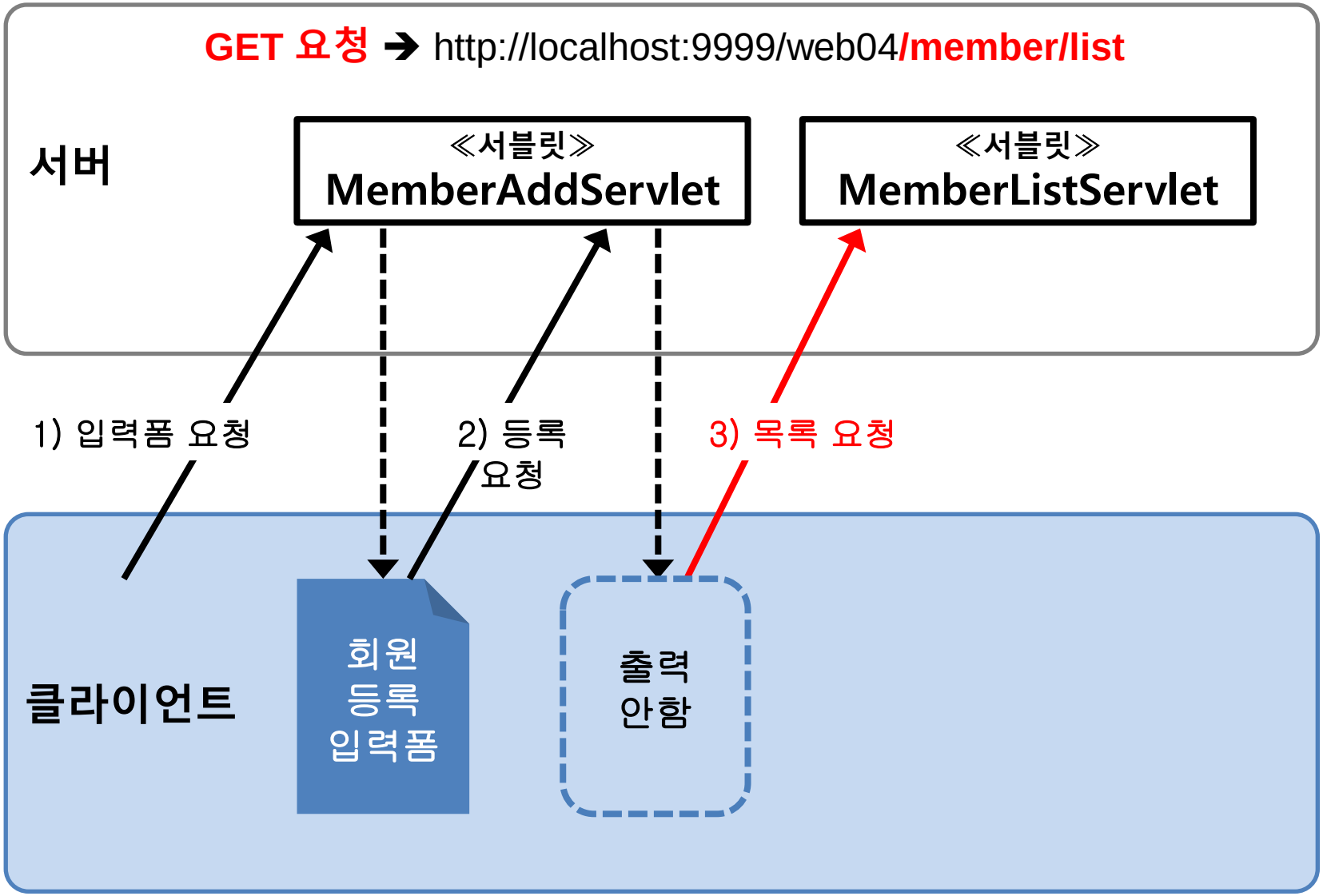
4.6 리다이렉트

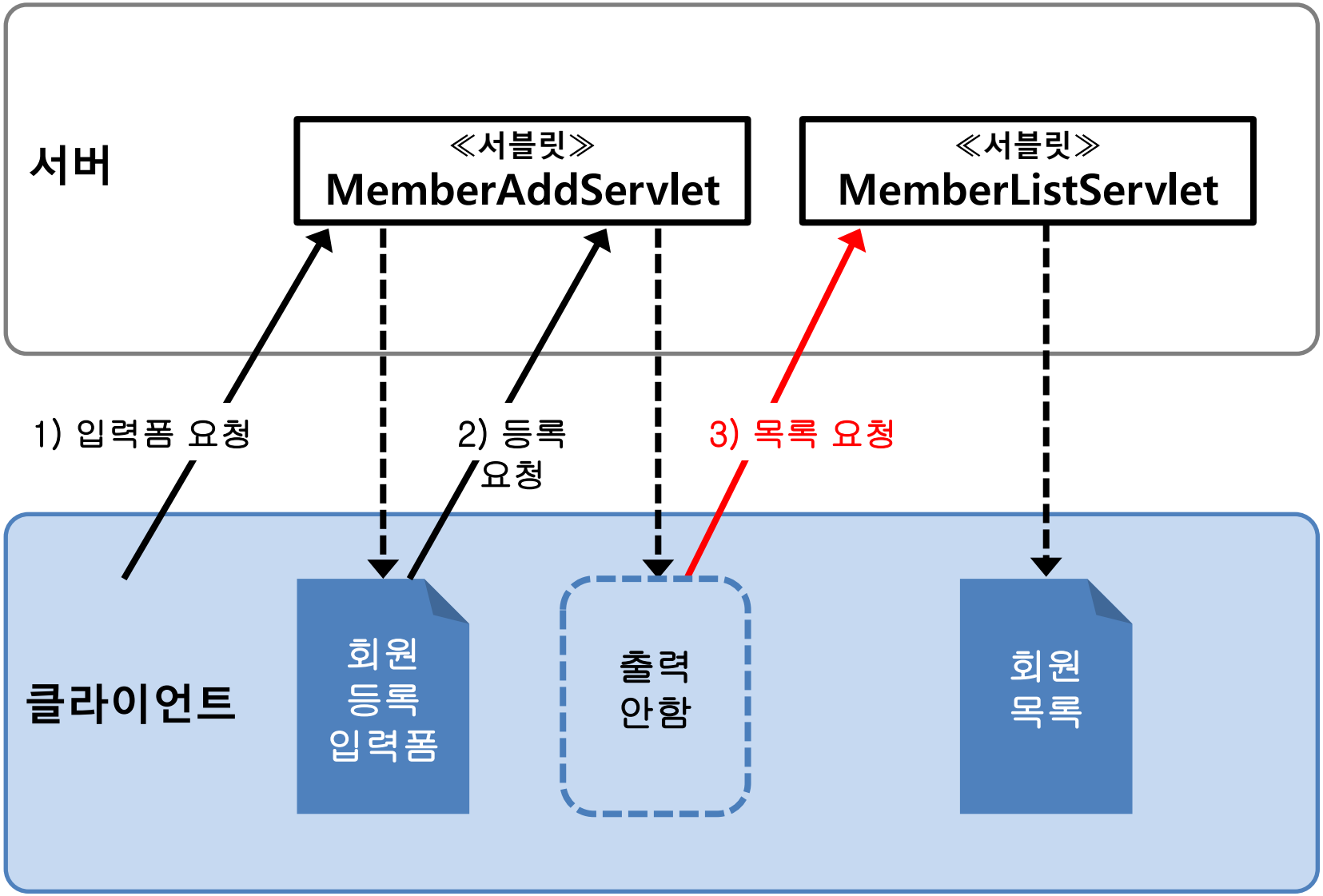

GET 요청 ➔ http://localhost:9999/web04/member/add
서버
≪서블릿≫
MemberAddServlet
≪서블릿≫
MemberListServlet
1) 입력폼 요청
클라이언트


서버
≪서블릿≫
MemberAddServlet
≪서블릿≫
MemberListServlet
1) 입력폼 요청
클라이언트
회원
등록
입력폼


POST 요청 ➔ http://localhost:9999/web04/member/add
서버
≪서블릿≫
MemberAddServlet
≪서블릿≫
MemberListServlet
1) 입력폼 요청
2) 등록 요청
클라이언트
회원 등록 입력폼


서버
≪서블릿≫
MemberAddServlet
≪서블릿≫
MemberListServlet
1) 입력폼 요청
2) 등록
요청
리다이렉트 응답
클라이언트
회원
등록
입력폼
출력
안함


GET 요청 ➔ http://localhost:9999/web04/member/list
서버
≪서블릿≫
MemberAddServlet
≪서블릿≫
MemberListServlet
1) 입력폼 요청
2) 등록 요청
3) 목록 요청
클라이언트
회원 등록 입력폼
출력 안함


서버
≪서블릿≫
MemberAddServlet
≪서블릿≫
MemberListServlet
1) 입력폼 요청
2) 등록
요청
3) 목록 요청
클라이언트
회원
등록
입력폼
출력
안함
회원
목록

리다이렉트 하는 방법?

```
response.sendRedirect("list");
```

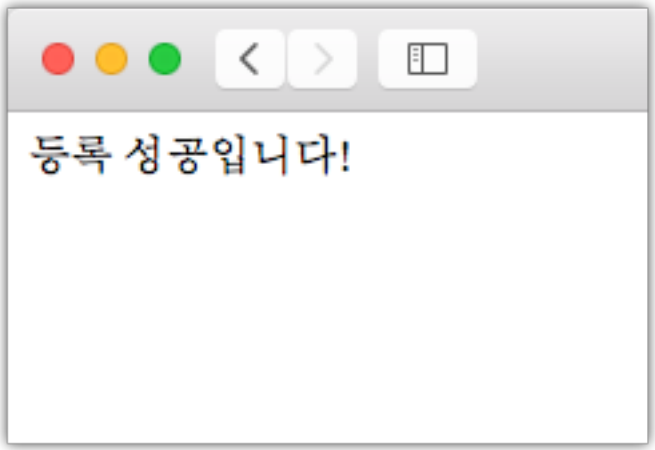
≪서블릿≫
MemberAddServlet

호출

doPost() { ... }

결과 응답

HTTP/1.1 302 Found
Server: Apache-Coyote/1.1
Location: http://localhost:9999/web04/member/list
Content-Length: 0
...

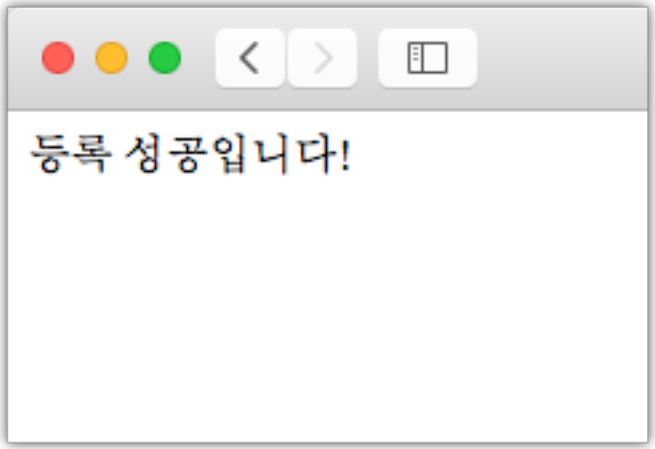


≪서블릿≫
MemberAddServlet

호출

doPost() { ... }

결과 응답



응답 상태 코드

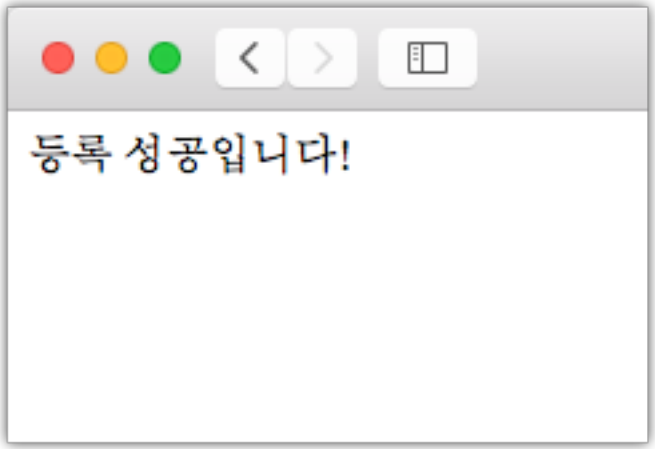
HTTP/1.1 302 Found
Server: Apache-Coyote/1.1
Location: http://localhost:9999/web04/member/list
Content-Length: 0
...

≪서블릿≫
MemberAddServlet

호출

doPost() { ... }

결과 응답



HTTP/1.1 302 Found
Server: Apache-Coyote/1.1
Location: http://localhost:9999/web04/member/list
Content-Length: 0
...

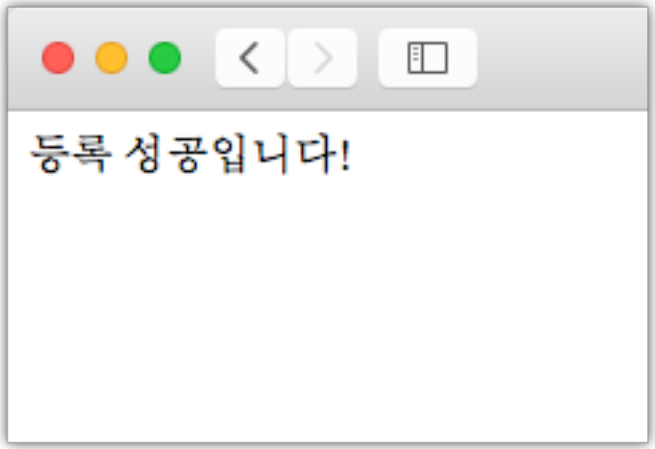
다시 요청해야 할 URL

≪서블릿≫
MemberAddServlet

호출

doPost() { ... }

결과 응답



HTTP/1.1 302 Found
Server: Apache-Coyote/1.1
Location: http://localhost:9999/web04/member/list
Content-Length: 0
...

웹브라우저에게 전송할 콘텐츠 크기

4.7 서블릿 초기화 매개변수

4.7 서블릿 초기화 매개변수

≪서블릿≫
MemberListServlet

DBMS 접속 정보
- URL
- Driver
- Username
- Password

4.7 서블릿 초기화 매개변수

≪서블릿≫
MemberListServlet

DBMS 접속 정보
- URL
- Driver
- Username
- Password

DBMS 정보 변경

4.7 서블릿 초기화 매개변수

≪서블릿≫
MemberListServlet

DBMS 접속 정보
- URL
- Driver
- Username
- Password

DBMS 정보 변경

⬇

서블릿 소스 변경

4.7 서블릿 초기화 매개변수

≪서블릿≫
MemberListServlet

DBMS 접속 정보
- URL
- Driver
- Username
- Password

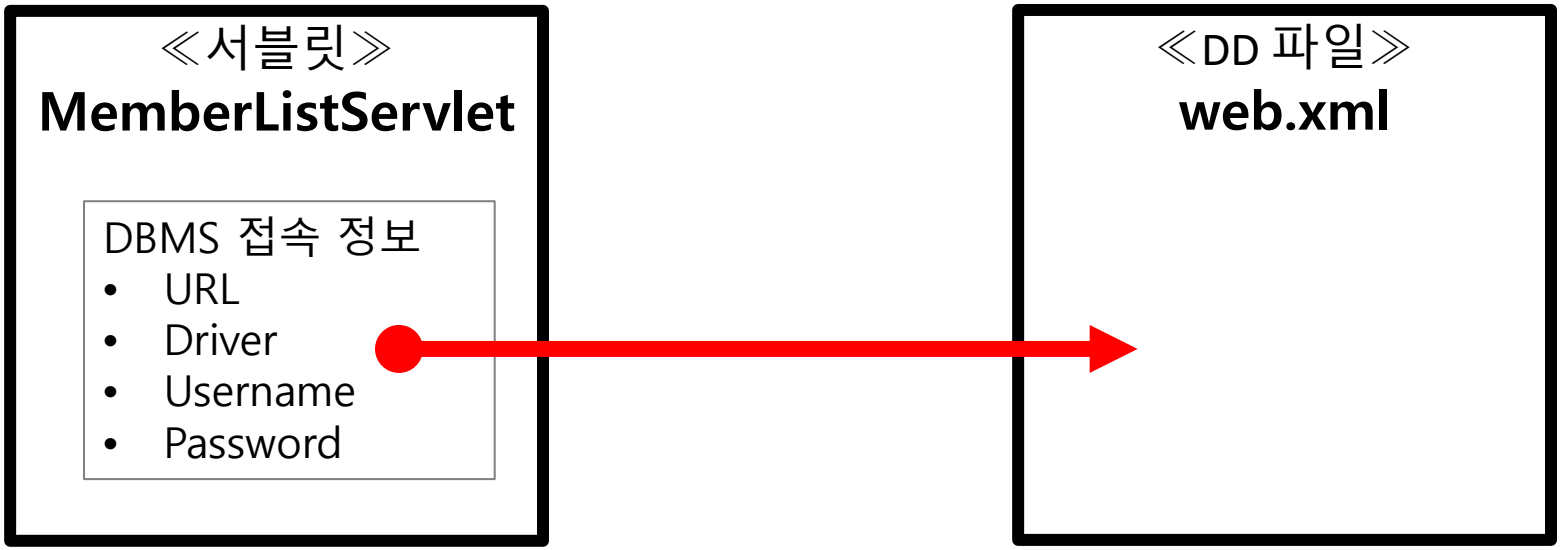
DBMS 정보 변경

서블릿 소스 변경

서버에 재배포

변경될 수 있는 값을 손쉽게 관리하기

4.7 서블릿 초기화 매개변수



- DD 파일 → Deployment Descriptor 파일

설정 방법?

4.7 서블릿 초기화 매개변수

web.xml

```
<servlet>
    …
    <init-param>
        <param-name>driver</param-name>
        <param-value>com.mysql.jdbc.Driver</param-value>
    </init-param>

    <init-param>
        <param-name>username</param-name>
        <param-value>study</param-value>
    </init-param>
    …
</servlet>
```

애노테이션으로 설정

@WebInitParam 애노테이션

```
@WebServlet(
  urlPatterns={"/member/update"},
   initParams={
    @WebInitParam(name="driver”, value="com.mysql.jdbc.Driver”),
    @WebInitParam(name=”username”, value=”study”),
    …
  }
)
public class MemberListServlet extends HttpServlet { … }
```

서블릿의
초기화 매개변수 값 꺼내기

4.7 서블릿 초기화 매개변수

≪DD 파일≫

web.xml

DBMS 접속 정보

- URL
- Driver
- Username
- Password

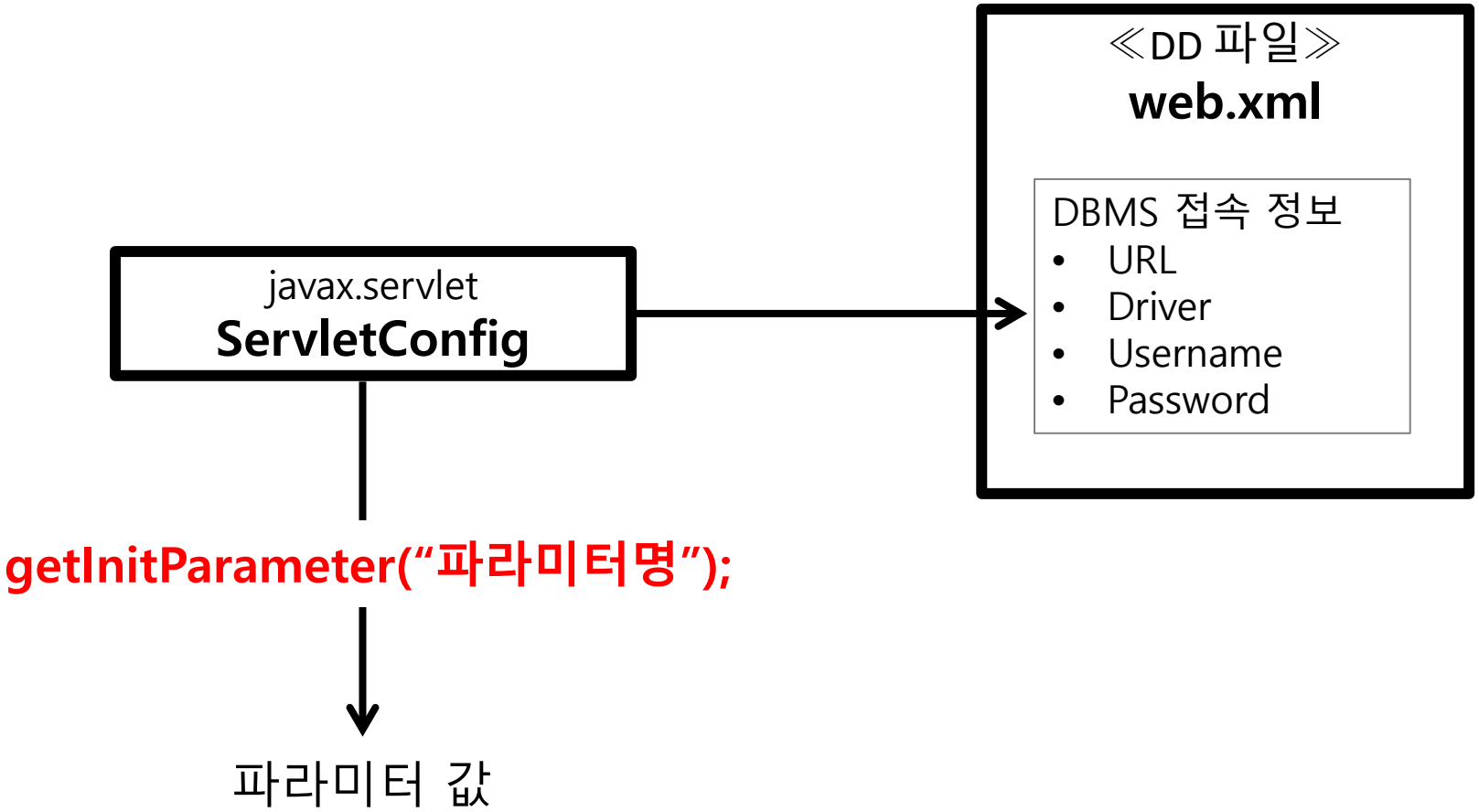
4.7 서블릿 초기화 매개변수

javax.servlet
ServletConfig

≪DD 파일≫
web.xml

DBMS 접속 정보
- URL
- Driver
- Username
- Password

4.7 서블릿 초기화 매개변수



4.8 컨텍스트 초기화 매개변수

4.8 컨텍스트 초기화 매개변수

web.xml

```
<web-app>
    ...
    <context-param>
        <param-name>driver</param-name>
        <param-value>com.mysql.jdbc.Driver</param-value>
    </context-param>

    <context-param>
        <param-name>username</param-name>
        <param-value>study</param-value>
    </context-param>
    ...
</web-app>
```

4.8 컨텍스트 초기화 매개변수

≪DD 파일≫

web.xml

DBMS 접속 정보

- URL
- Driver
- Username
- Password

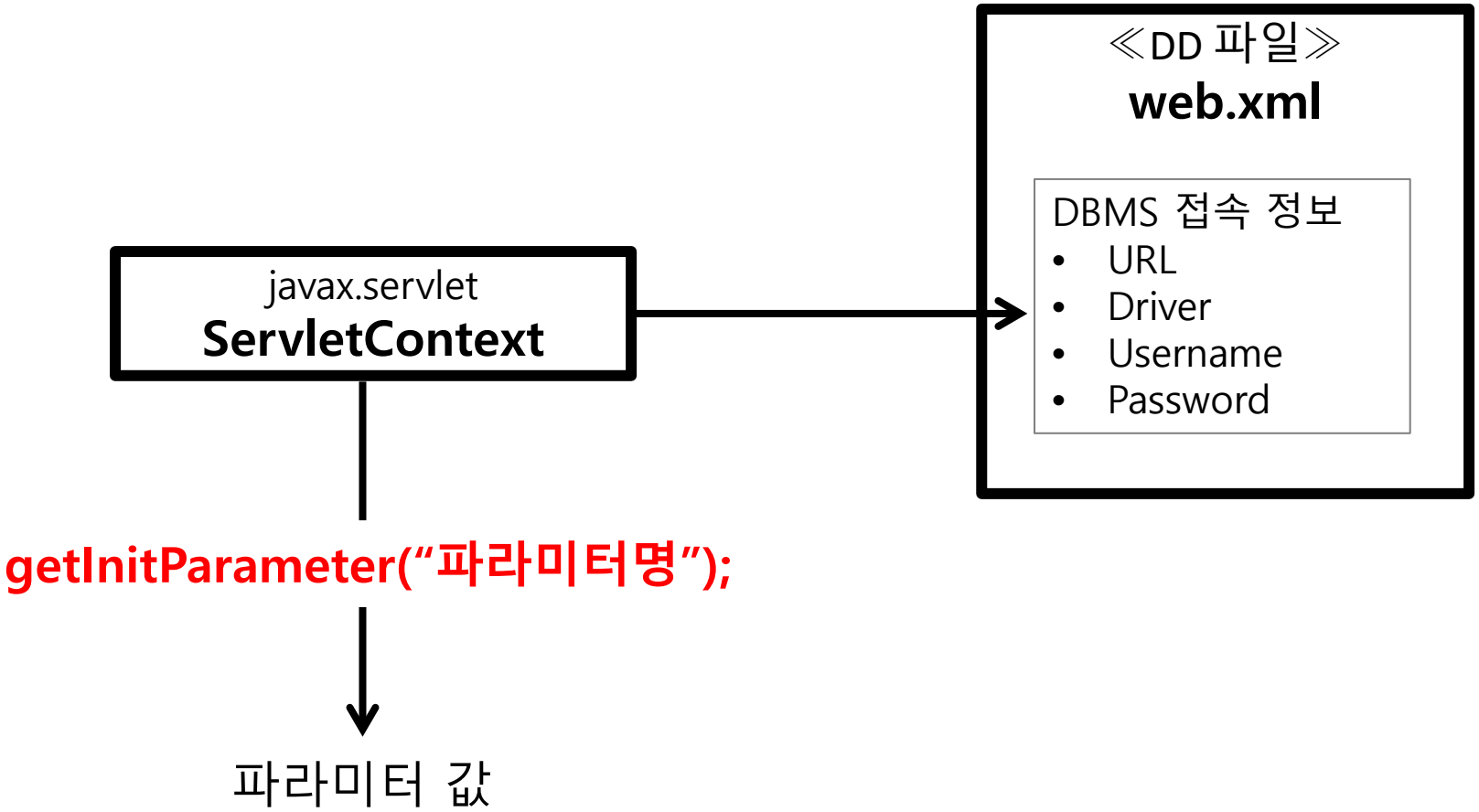
4.8 컨텍스트 초기화 매개변수

javax.servlet
ServletContext

≪DD 파일≫
web.xml

DBMS 접속 정보
- URL
- Driver
- Username
- Password

4.8 컨텍스트 초기화 매개변수



서블릿 초기화 매개변수
VS
컨텍스트 초기화 매개변수

해당 서블릿만
사용 가능

서블릿 초기화 매개변수
VS
컨텍스트 초기화 매개변수

해당 서블릿만
사용 가능

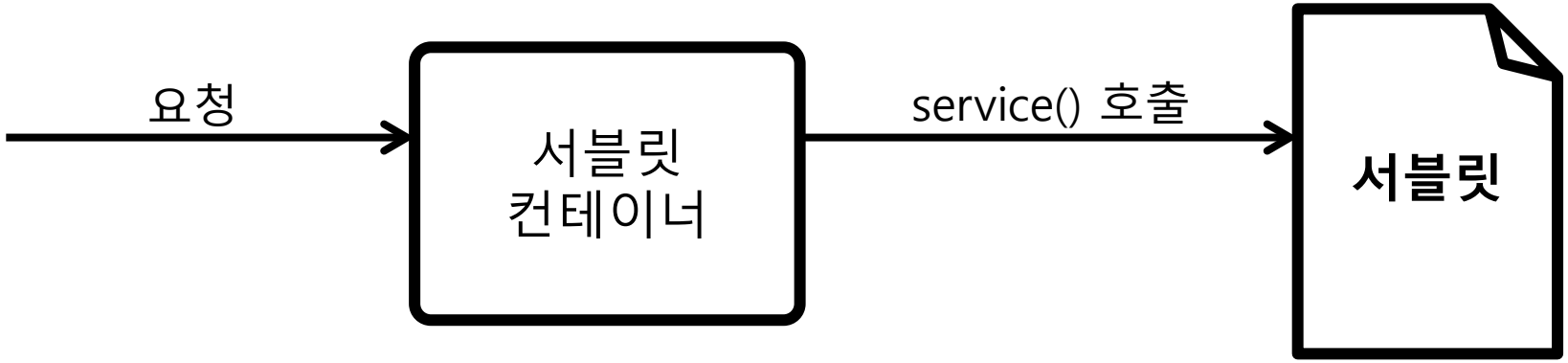
서블릿 초기화 매개변수
VS
컨텍스트 초기화 매개변수

모든 서블릿이
사용 가능

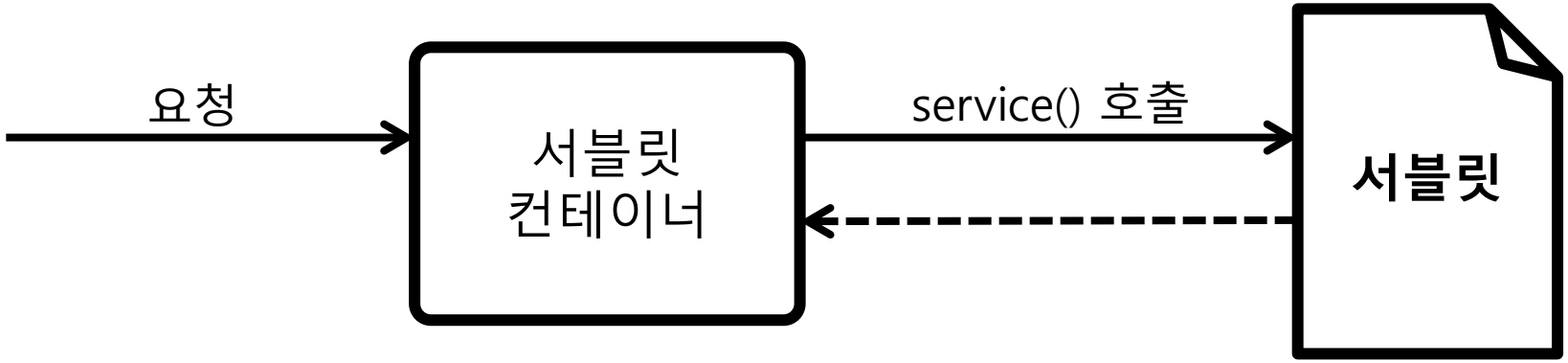
4.9 필터 사용하기

4.9 필터 사용하기

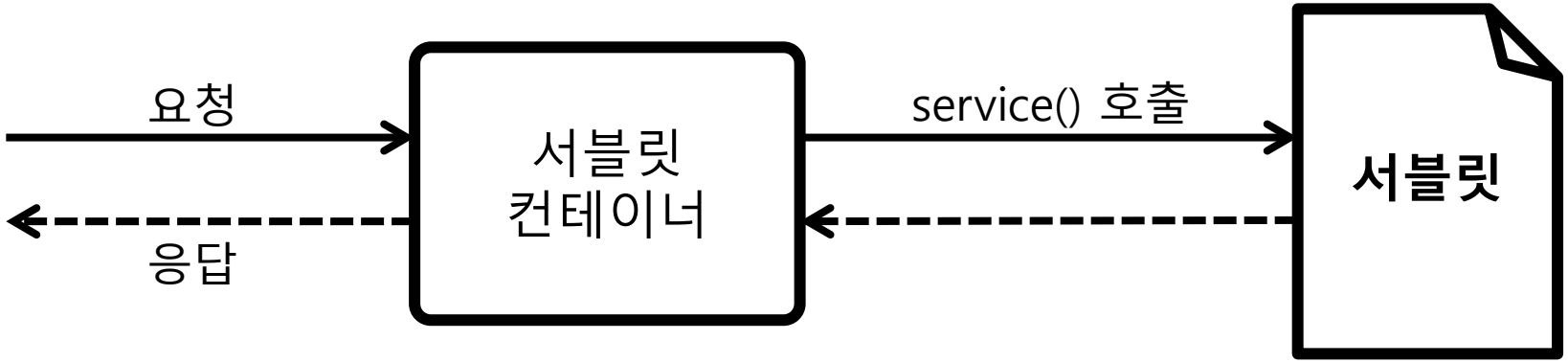
4.9 필터 사용하기



4.9 필터 사용하기

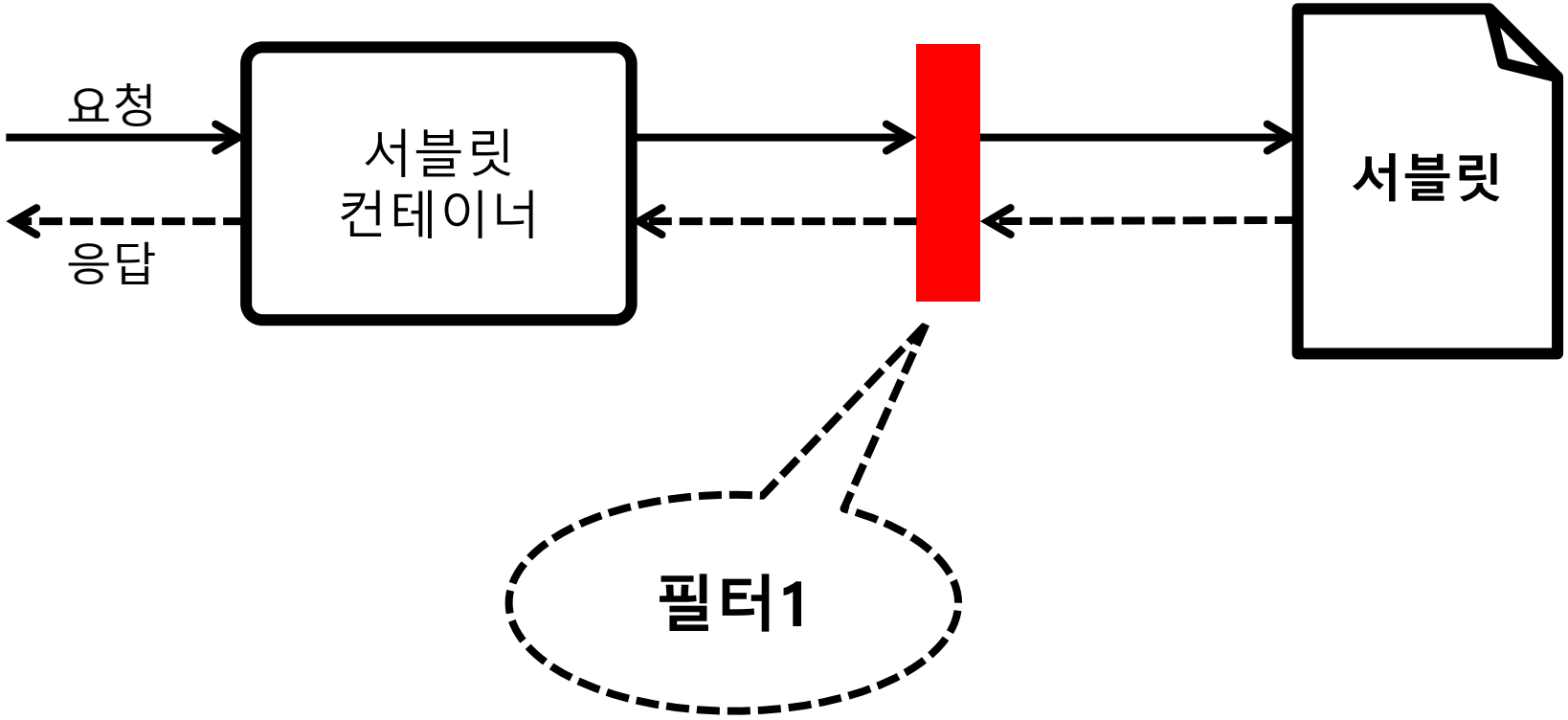


4.9 필터 사용하기

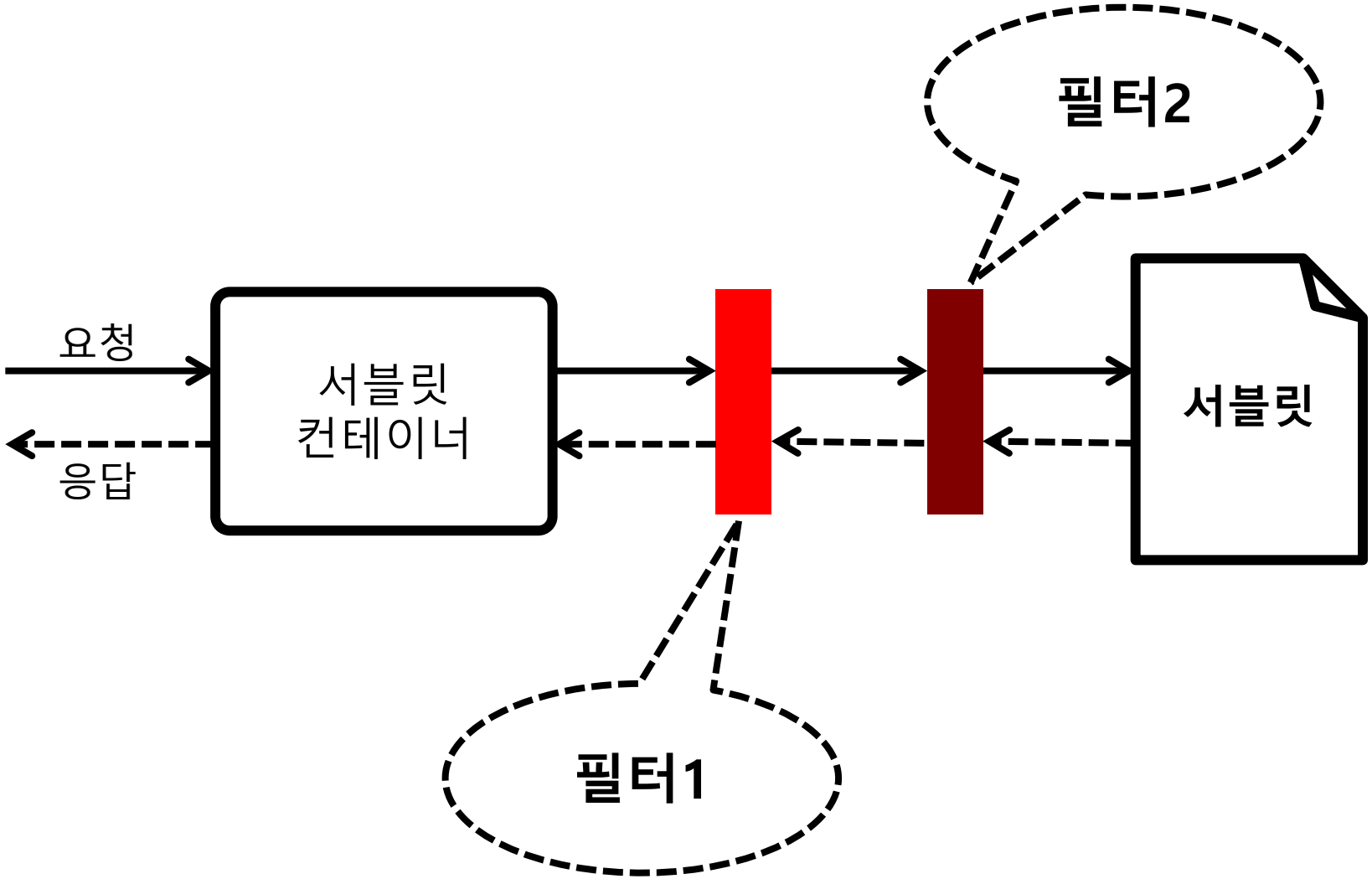


필터 넣기

4.9 필터 사용하기

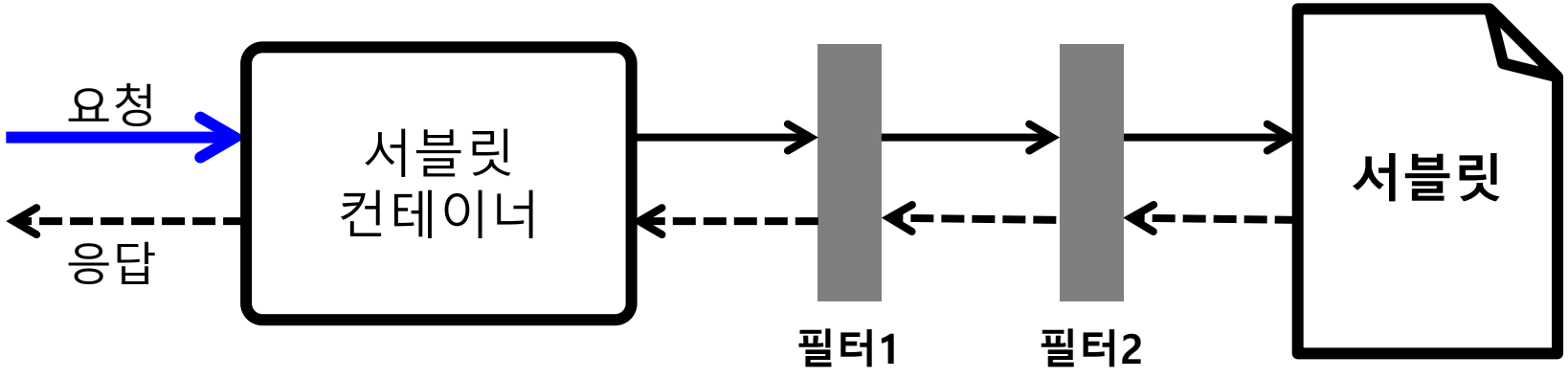


4.9 필터 사용하기

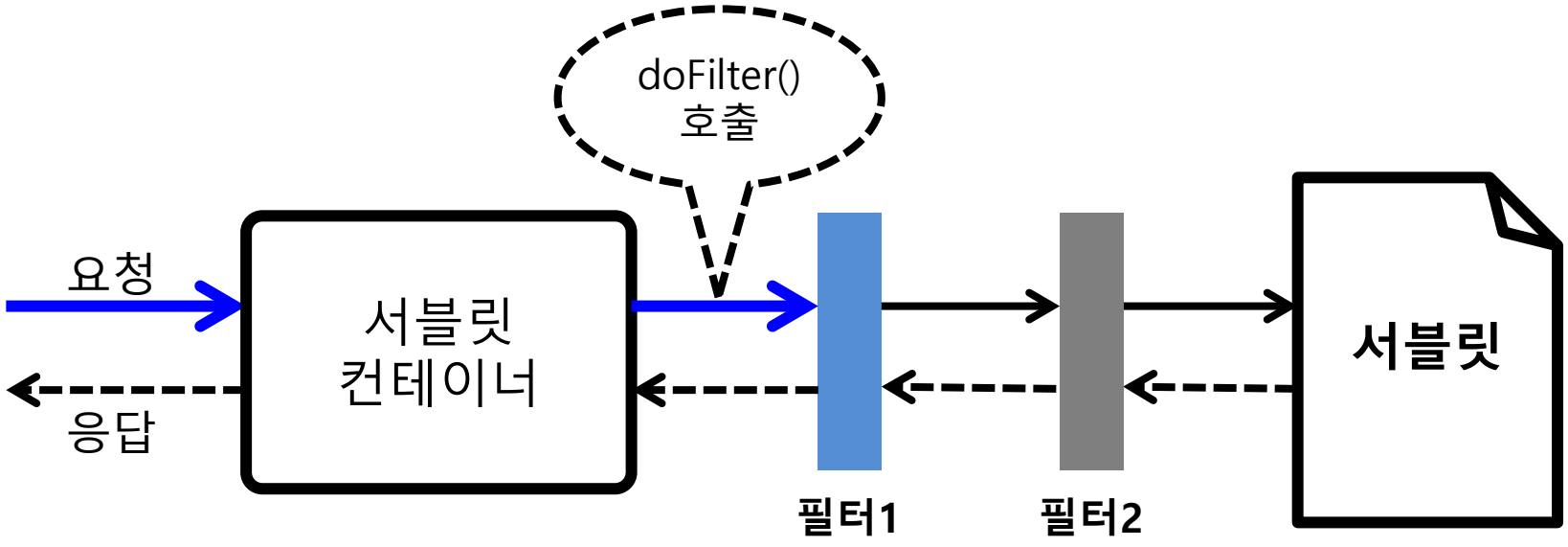


필터 실행!

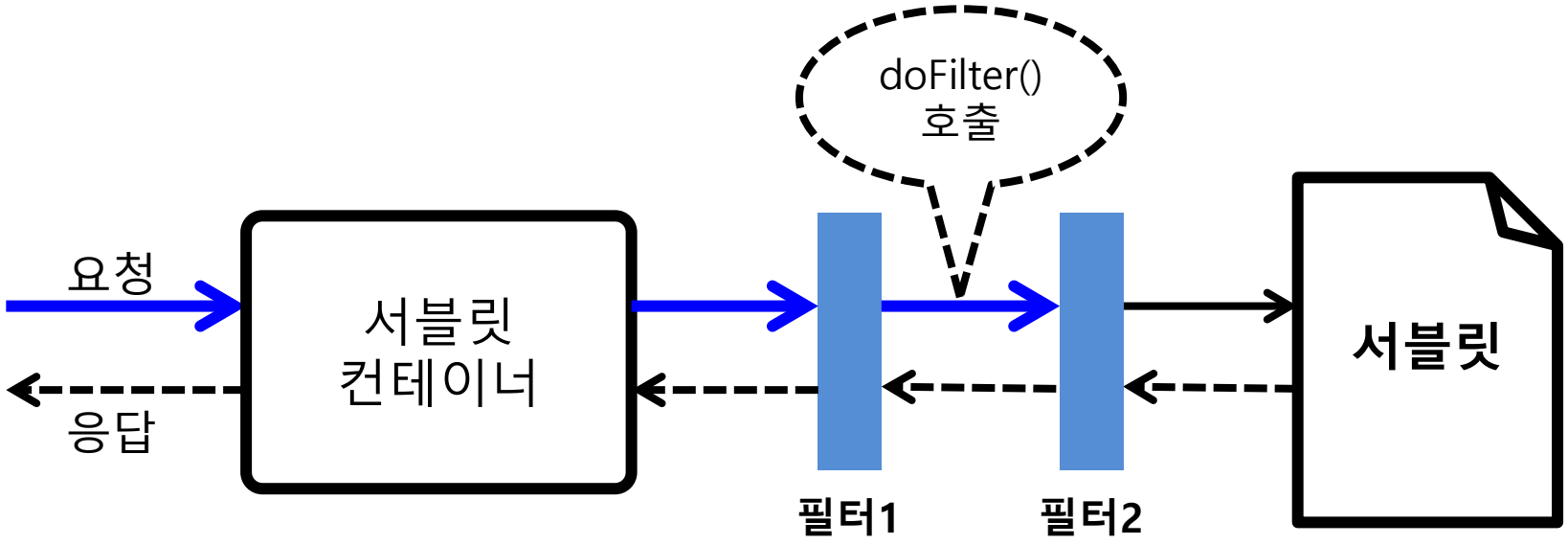
4.9 필터 사용하기



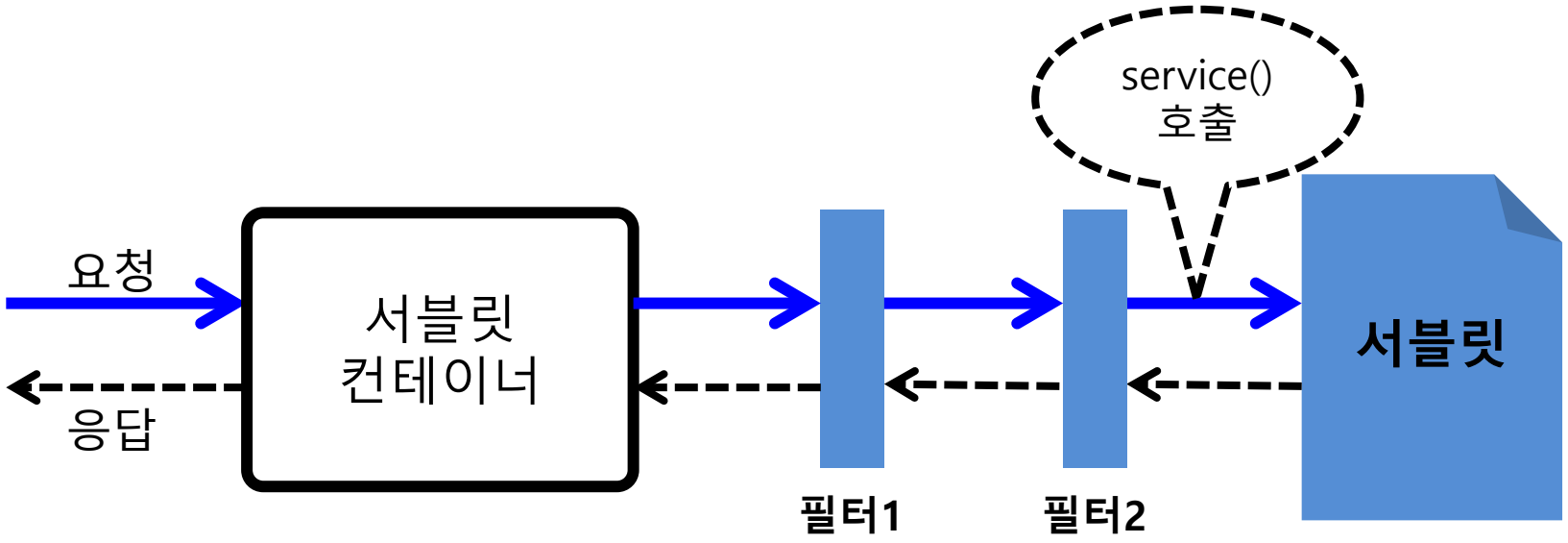
4.9 필터 사용하기



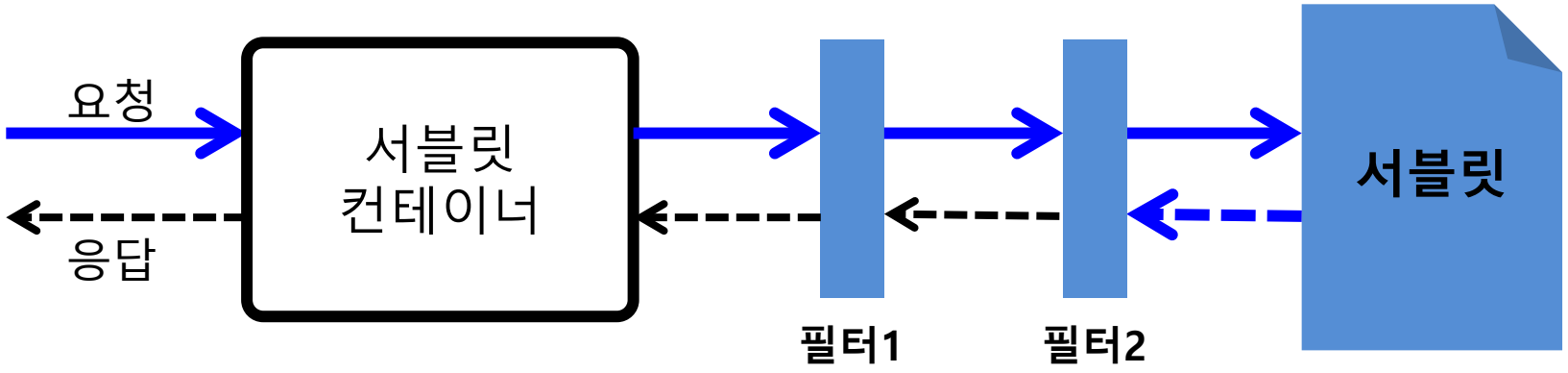
4.9 필터 사용하기



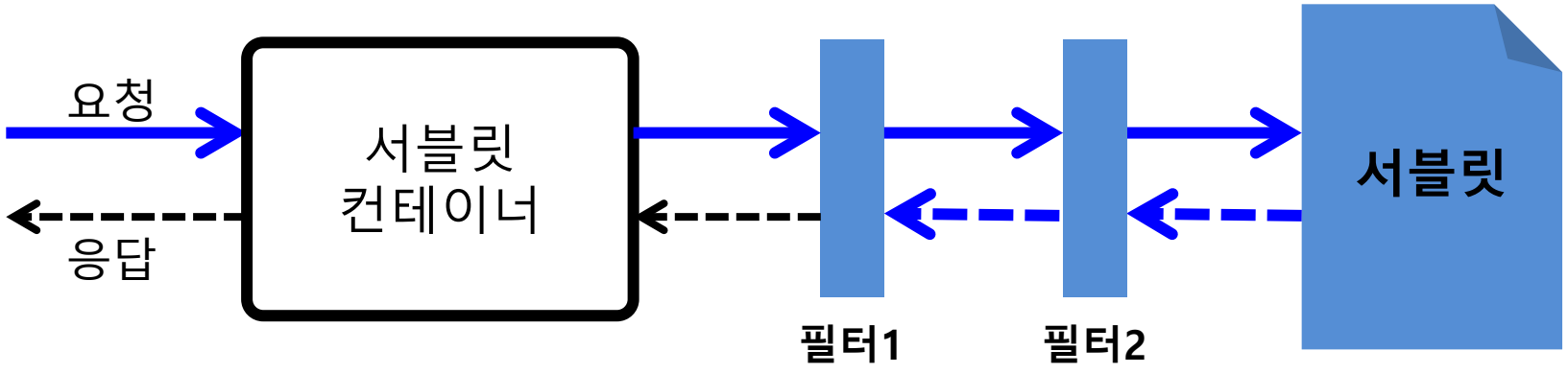
4.9 필터 사용하기



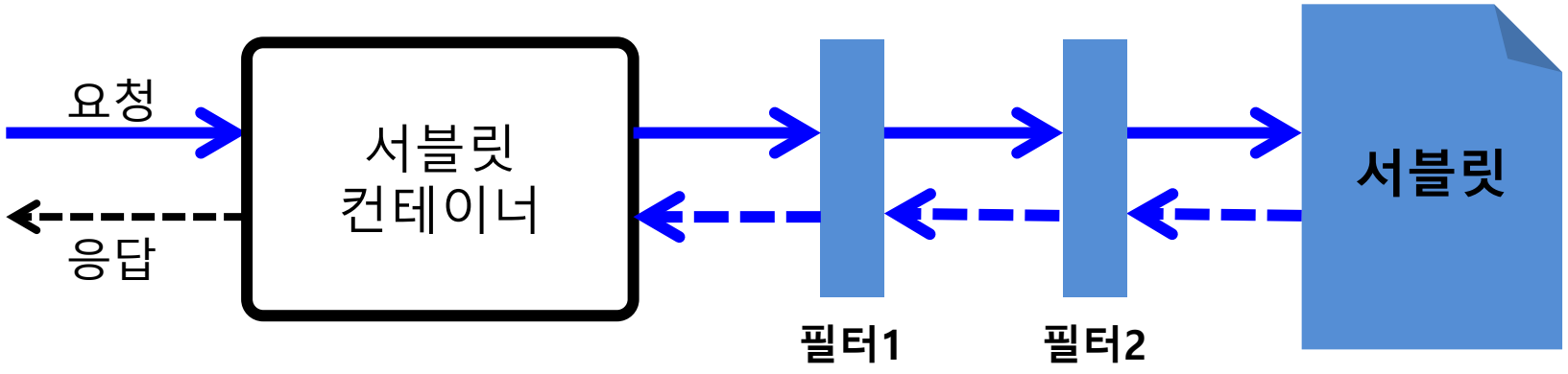
4.9 필터 사용하기



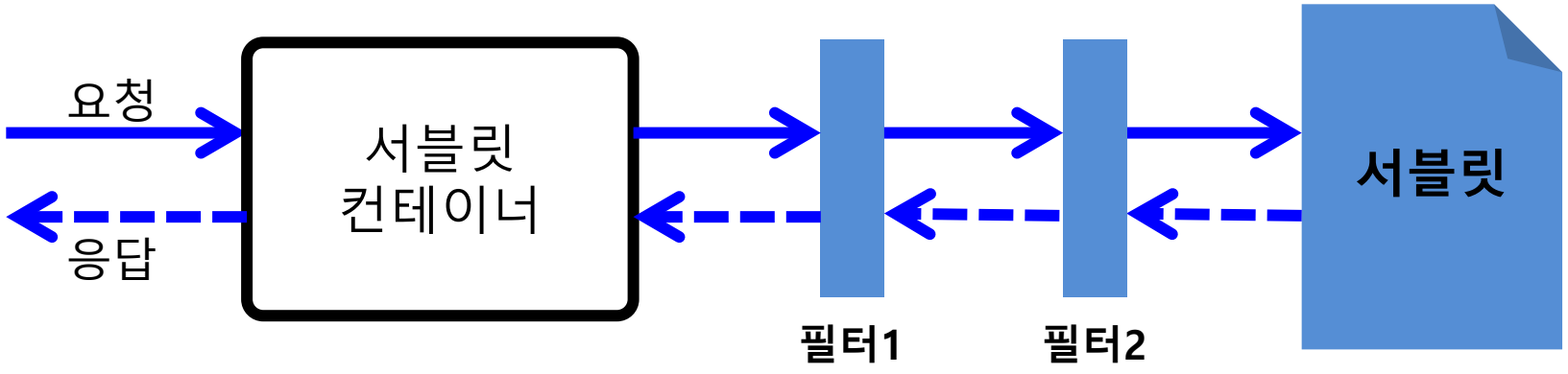
4.9 필터 사용하기



4.9 필터 사용하기



4.9 필터 사용하기



필터의 용도

로그 출력

로그 출력

사용자 인증 및 권한 검사

로그 출력

사용자 인증 및 권한 검사

암호화 및 복호화

로그 출력

사용자 인증 및 권한 검사

암호화 및 복호화

데이터 압축 및 해제

로그 출력

사용자 인증 및 권한 검사

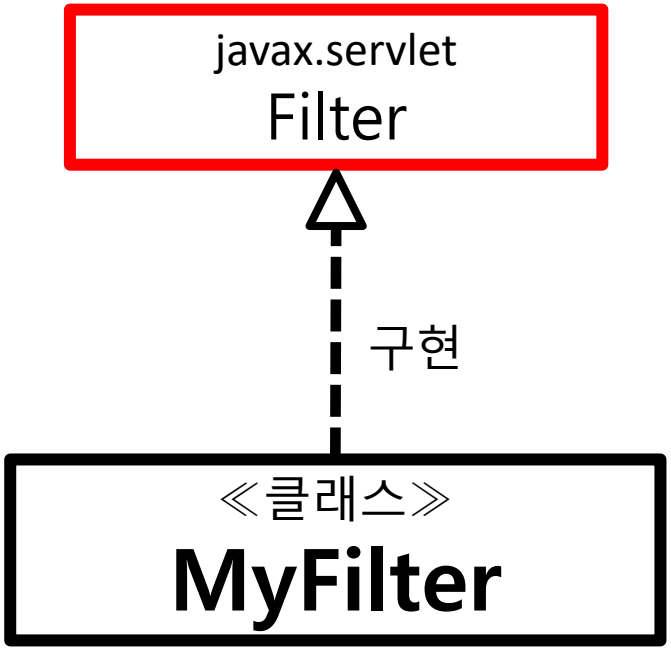
암호화 및 복호화

데이터 압축 및 해제

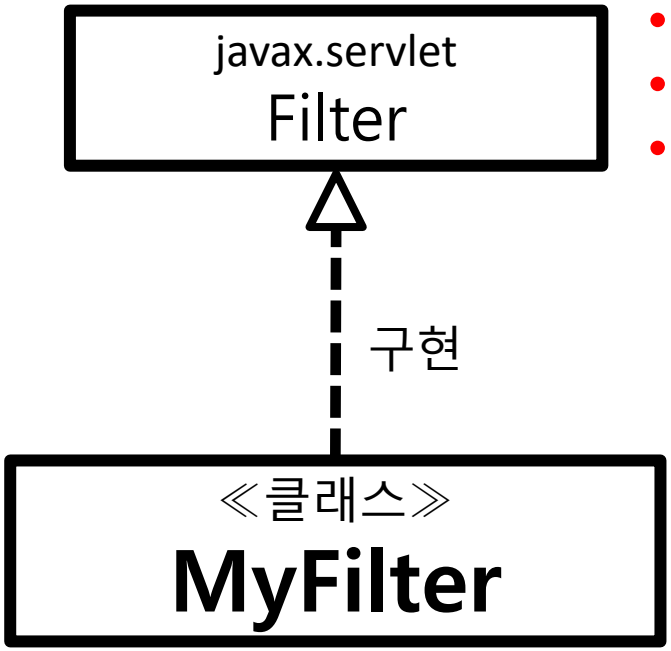
이미지 변환 등

필터 만들기

4.9 필터 사용하기



4.9 필터 사용하기



- **init() { … }**
- **destroy() { … }**
- **doFilter() { … }**

필터 적용

web.xml

```
<servlet>
    …
    <filter>
        <filter-name>필터 이름</filter-name>
        <filter-class>필터 클래스</filter-class>
    </filter>

    <filter-mapping>
        <filter-name>필터 이름</filter-name>
        <url-pattern>URL 패턴</url-pattern>
    </filter-mapping>
    …
</servlet>
```

web.xml

```
<servlet>
    …
    <filter>
        <filter-name>필터 이름</filter-name>
        <filter-class>필터 클래스</filter-class>
    </filter>

    <filter-mapping>
        <filter-name>필터 이름</filter-name>
        <url-pattern>URL 패턴</url-pattern>
    </filter-mapping>
    …
</servlet>
```

애노테이션으로 필터 지정

4.9 필터 사용하기

```
@WebFilter( urlPatterns="/*")
```

4.9 필터 사용하기

```
@WebFilter( urlPatterns="/*")

@WebFilter(
   urlPatterns="/*",
   initParams={
      @WebInitParam(
    name="encoding”, value="UTF-8")
   })
```