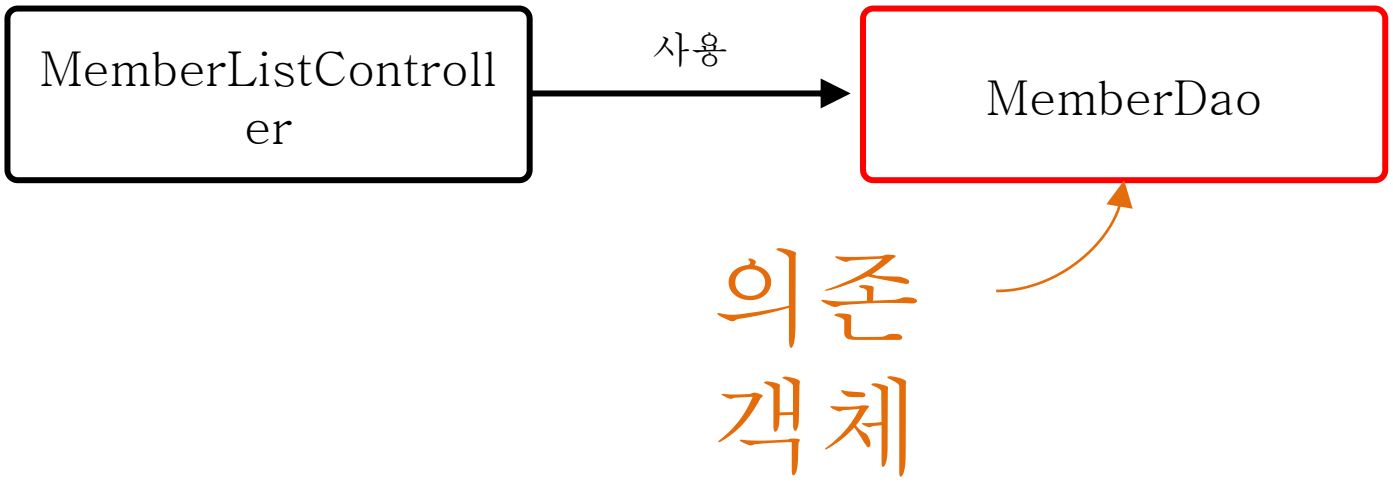
6.3 DI를 이용한 빈 의존성 관리

6.3 DI를 이용한 빈 의존성 관리

“의존 객체”
특정 작업을 수행할 때 사용하는 객체

6.3 DI를 이용한 빈 의존성 관리

"의존 객체"
특정 작업을 수행할 때 사용하는 객체



6.3 DI를 이용한 빈 의존성 관리

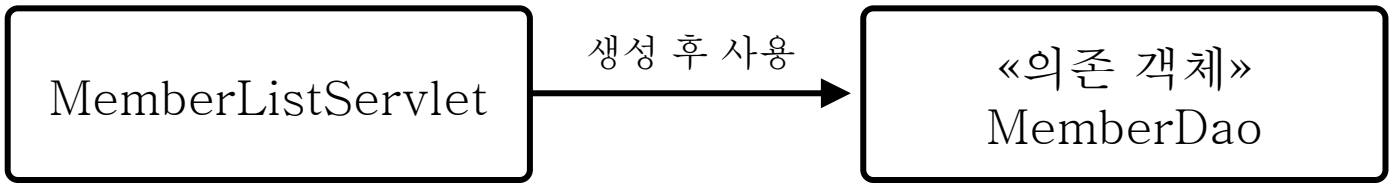
“의존 객체”
특정 작업을 수행할 때 사용하는 객체


```
public class MemberListController implements Controller {
  public String execute(Map<String, Object> model) throws Exception {
    MemberDao memberDao = (MemberDao)model.get("memberDao");
    model.put("members", memberDao.selectList());
    return "/member/MemberList.jsp";
  }
}
```

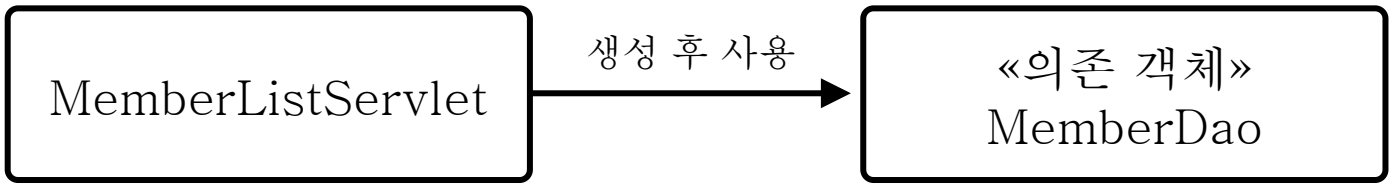
의존 객체 관리

1) 사용할 때 마다 의존 객체 생성하기

사용할 때 마다 의존 객체 생성하기

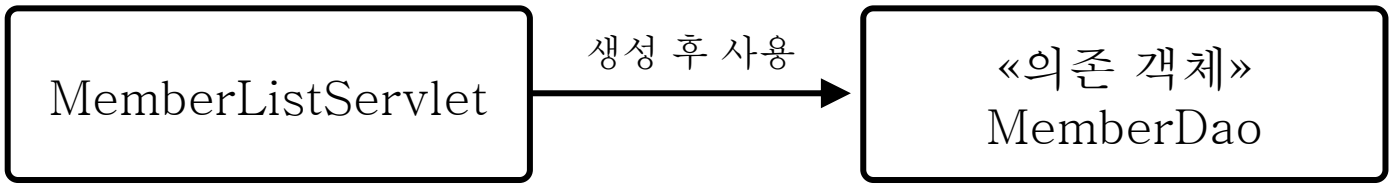


사용할 때 마다 의존 객체 생성하기




```
public class MemberListServlet extends HttpServlet {
  public void doGet(…) throws ServletException, IOException {
    …
    MemberDao memberDao = new MemberDao();
    memberDao.setConnection(conn);
    request.setAttribute("members", memberDao.selectList());
```

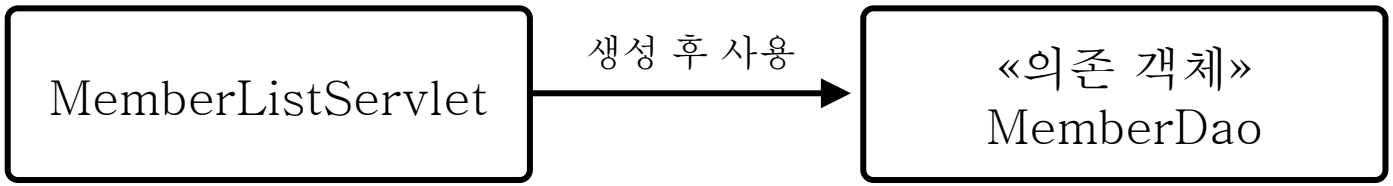
사용할 때 마다 의존 객체 생성하기




```
public class MemberListServlet extends HttpServlet {
  public void doGet(…) throws ServletException, IOException {
    …
    MemberDao memberDao = new MemberDao();
    memberDao.setConnection(conn);
    request.setAttribute("members", memberDao.selectList());
```

의존 객체 생성

사용할 때 마다 의존 객체 생성하기

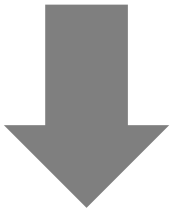



```
public class MemberListServlet extends HttpServlet {
  public void doGet(…) throws ServletException, IOException {
    …
    MemberDao memberDao = new MemberDao();
    memberDao.setConnection(conn);
    request.setAttribute("members", memberDao.selectList());
```

의존 객체 생성

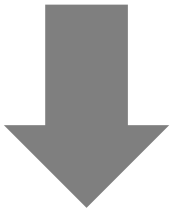
객체 사용

사용할 때 마다 의존 객체 생성하기



문제점

사용할 때 마다 의존 객체 생성하기



문제점

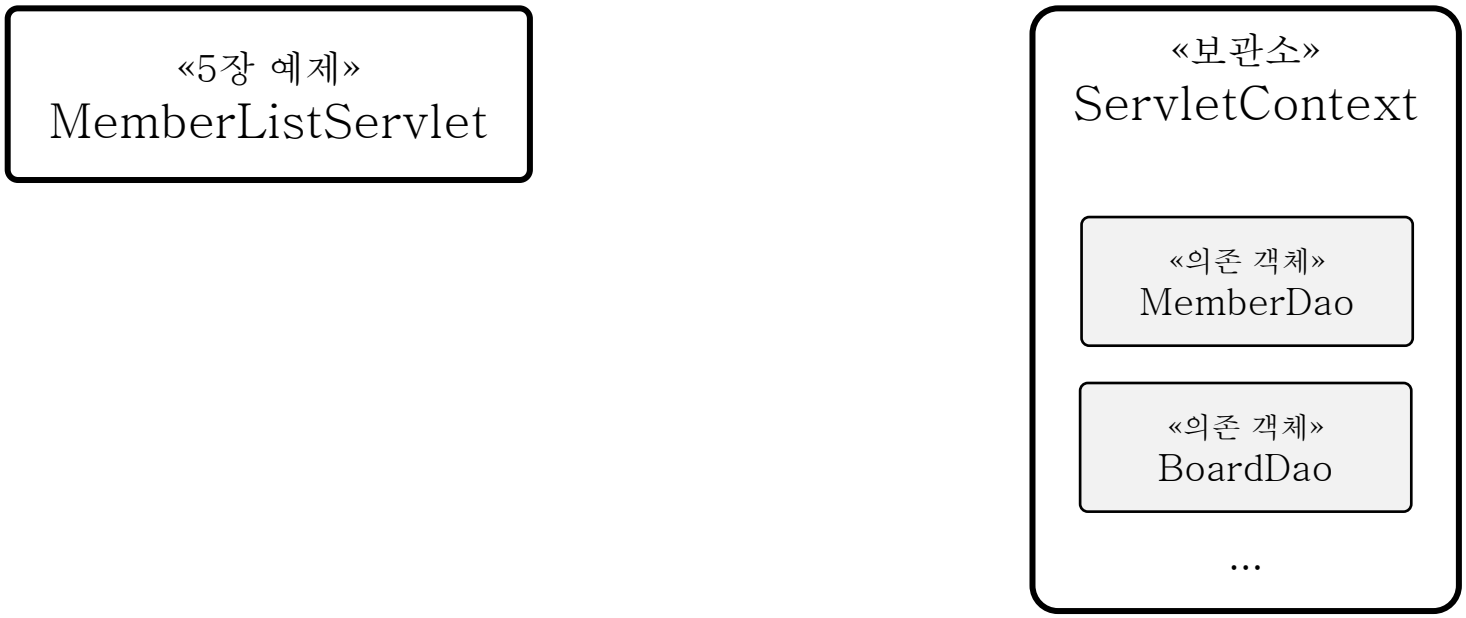
- 많은 가비지(garbage) 생성
- 실행 시간 지연

의존 객체 관리

2) 의존 객체를 미리 생성해 두었다가 필요할 때 꺼내쓰기

6.3 DI를 이용한 빈 의존성 관리

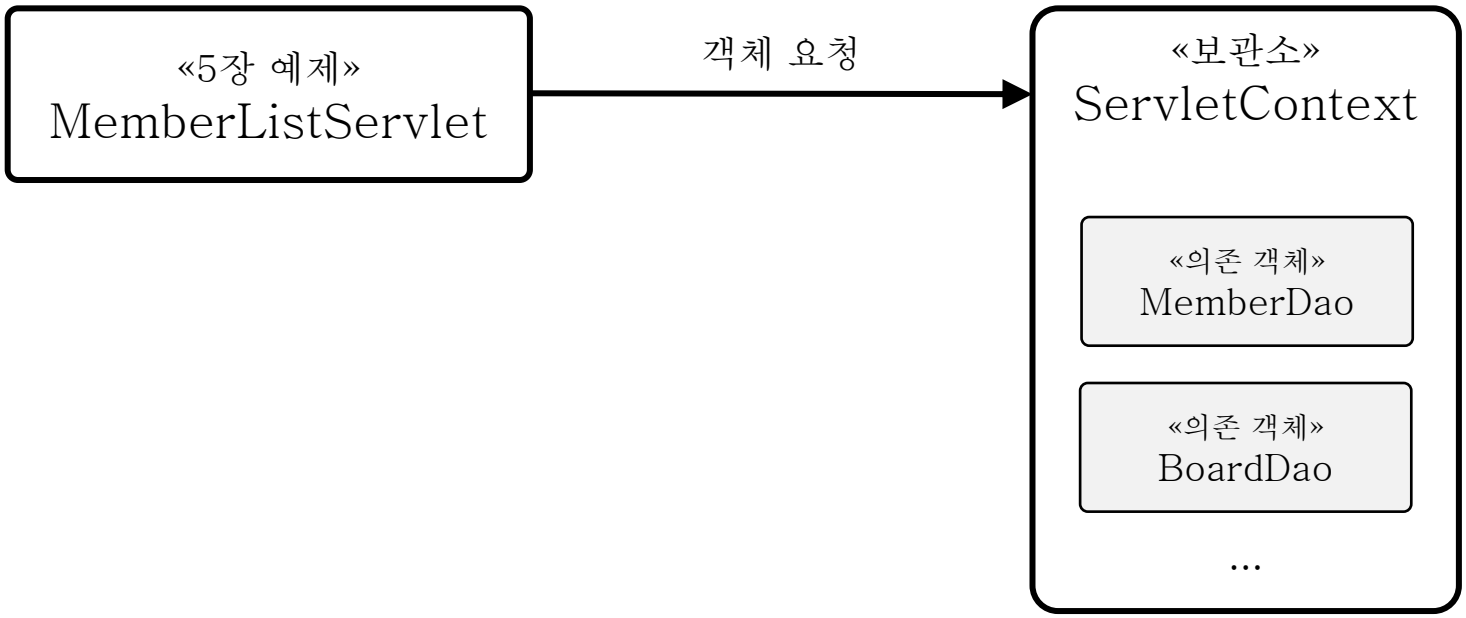
의존 객체를 미리 생성해 두었다가 필요할 때 꺼내 쓰기




```
public void doGet(…) throws ServletException, IOException {
 try {
   ServletContext sc = this.getServletContext();
   MemberDao memberDao = (MemberDao)sc.getAttribute("memberDao");
```

6.3 DI를 이용한 빈 의존성 관리

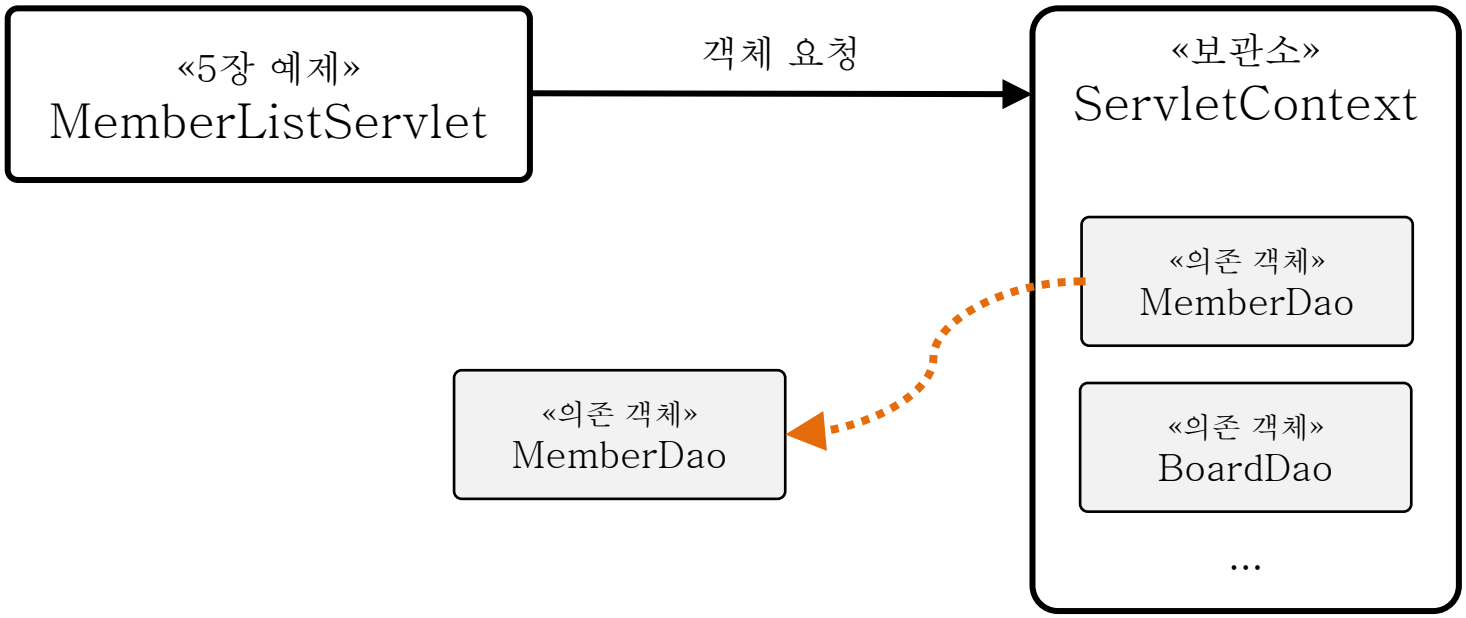
의존 객체를 미리 생성해 두었다가 필요할 때 꺼내 쓰기




```
public void doGet(…) throws ServletException, IOException {
 try {
   ServletContext sc = this.getServletContext();
   MemberDao memberDao = (MemberDao)sc.getAttribute("memberDao");
```

6.3 DI를 이용한 빈 의존성 관리

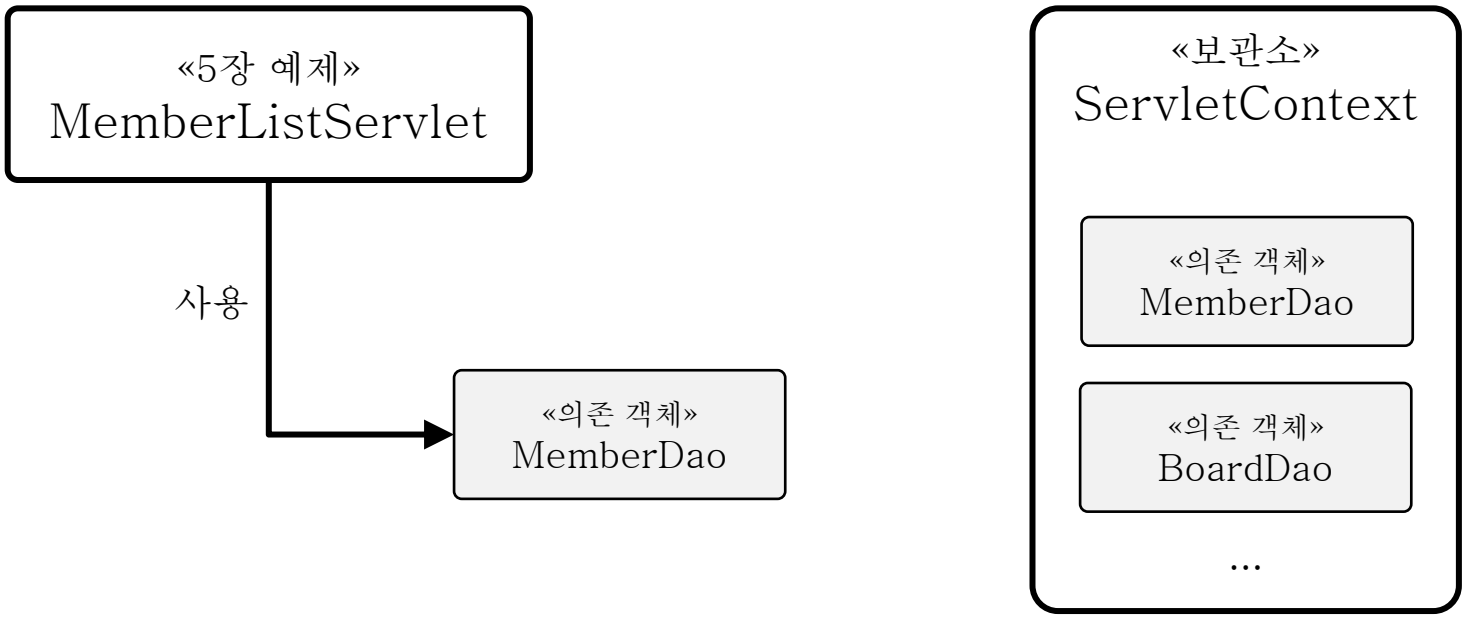
의존 객체를 미리 생성해 두었다가 필요할 때 꺼내 쓰기




```
public void doGet(…) throws ServletException, IOException {
 try {
   ServletContext sc = this.getServletContext();
   MemberDao memberDao = (MemberDao)sc.getAttribute("memberDao");
```

6.3 DI를 이용한 빈 의존성 관리

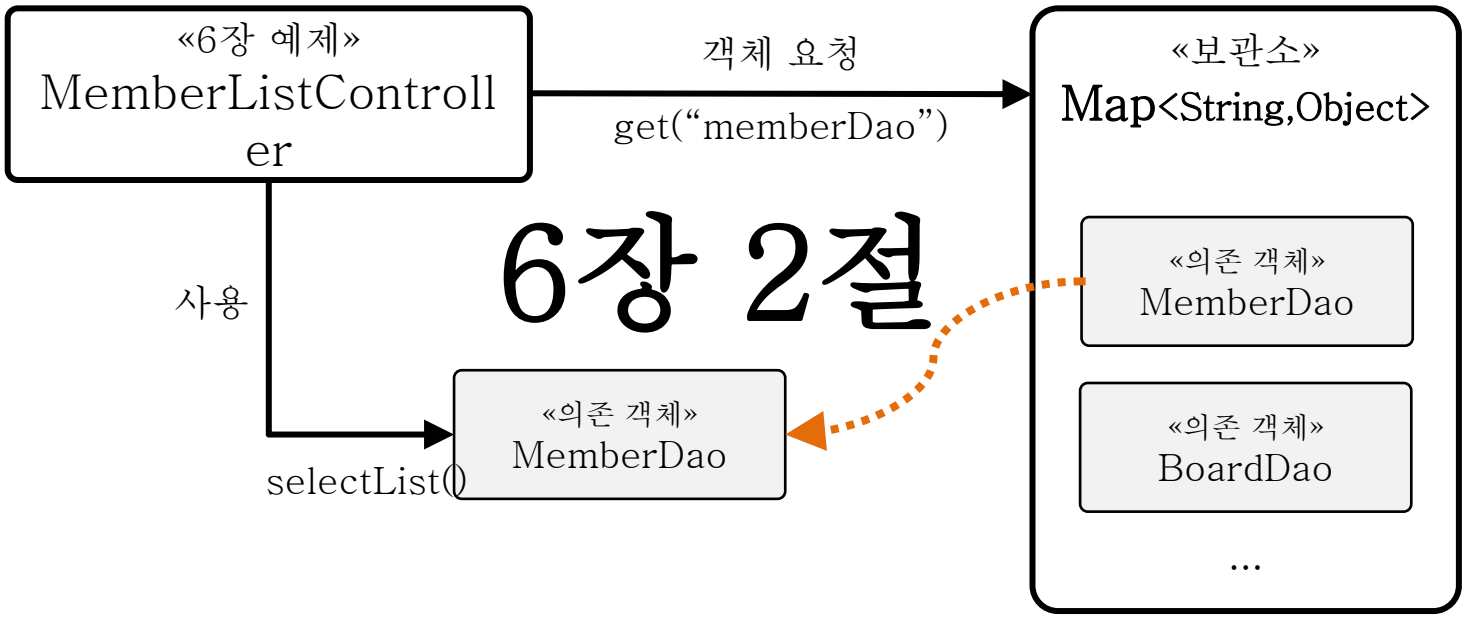
의존 객체를 미리 생성해 두었다가 필요할 때 꺼내 쓰기




```
public void doGet(…) throws ServletException, IOException {
 try {
   ServletContext sc = this.getServletContext();
   MemberDao memberDao = (MemberDao)sc.getAttribute("memberDao");
   request.setAttribute("members", memberDao.selectList());
```

6.3 DI를 이용한 빈 의존성 관리

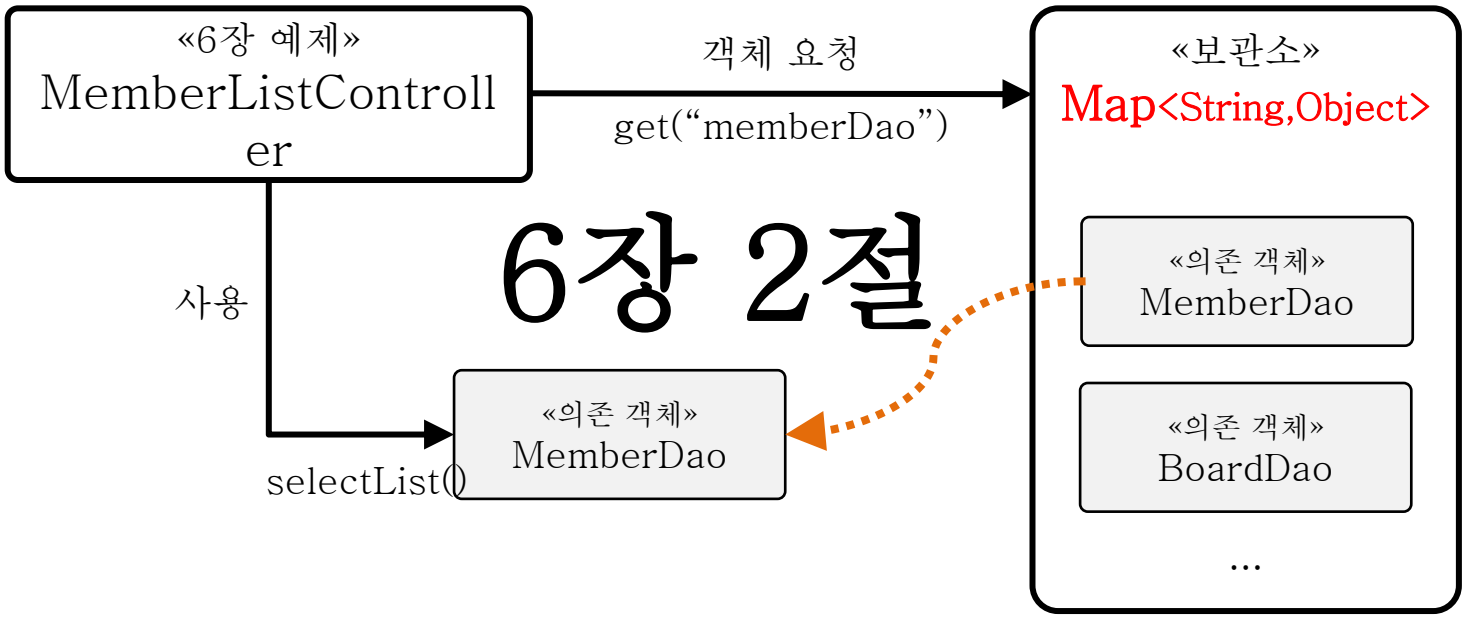
의존 객체를 미리 생성해 두었다가 필요할 때 꺼내 쓰기




```
public String execute(Map<String, Object> model) throws Exception {
  MemberDao memberDao = (MemberDao)model.get("memberDao");
  model.put("members", memberDao.selectList());
  return "/member/MemberList.jsp";
}
```

6.3 DI를 이용한 빈 의존성 관리

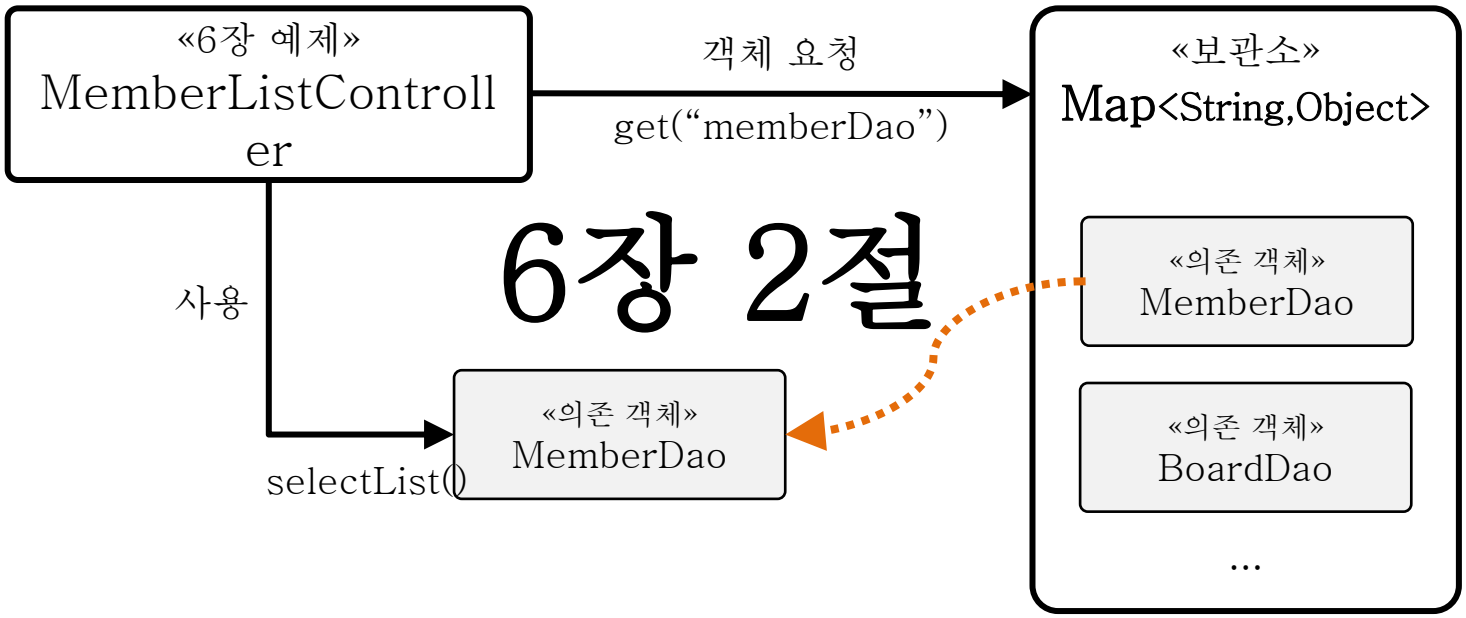
의존 객체를 미리 생성해 두었다가 필요할 때 꺼내 쓰기




```
public String execute(Map<String, Object> model) throws Exception {
  MemberDao memberDao = (MemberDao)model.get("memberDao");
  model.put("members", memberDao.selectList());
  return "/member/MemberList.jsp";
}
```

6.3 DI를 이용한 빈 의존성 관리

의존 객체를 미리 생성해 두었다가 필요할 때 꺼내 쓰기




```
public String execute(Map<String, Object> model) throws Exception {
  MemberDao memberDao = (MemberDao)model.get("memberDao");
  model.put("members", memberDao.selectList());
  return "/member/MemberList.jsp";
}
```

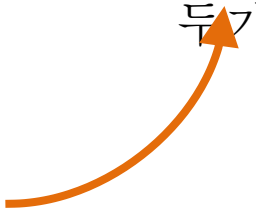
의존 객체 관리

3) 필요한 의존 객체를 사용 전에 미리 주입해 두기

6.3 DI를 이용한 빈 의존성 관리

의존 객체 관리

3) 필요한 의존 객체를 사용 전에 미리 주입해 두기



6장 3절의
핵심 내용

의존 객체 관리

3) 필요한 의존 객체를 사용 전에 미리 주입해 두기

6장 3절의 핵심 내용
“의존성 주입(Dependency Injection; DI)”

6.3 DI를 이용한 빈 의존성 관리

필요한 의존 객체를 사용 전에 미리 주입해 두기

«빈 관리자»
ContextLoaderListener

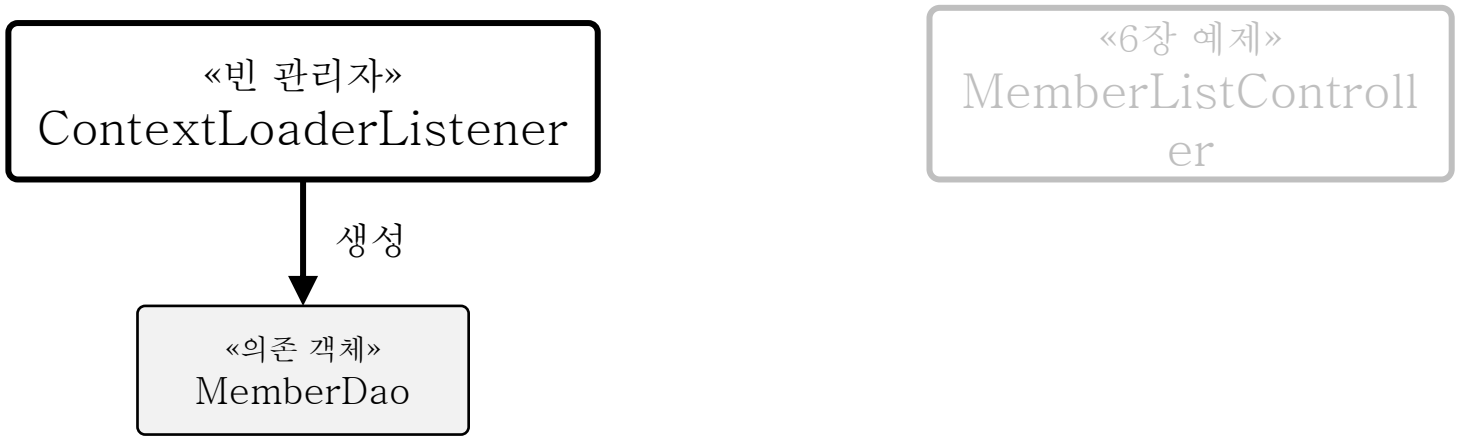
«6장 예제»
MemberListControll
er

«의존 객체»
MemberDao

```
public class ContextLoaderListener implements ServletContextListener {
  public void contextInitialized(ServletContextEvent event) {
    …
    MemberDao memberDao = new MemberDao();
    …
    sc.setAttribute("/member/list.do",
        new MemberListController().setMemberDao(memberDao));
```

6.3 DI를 이용한 빈 의존성 관리

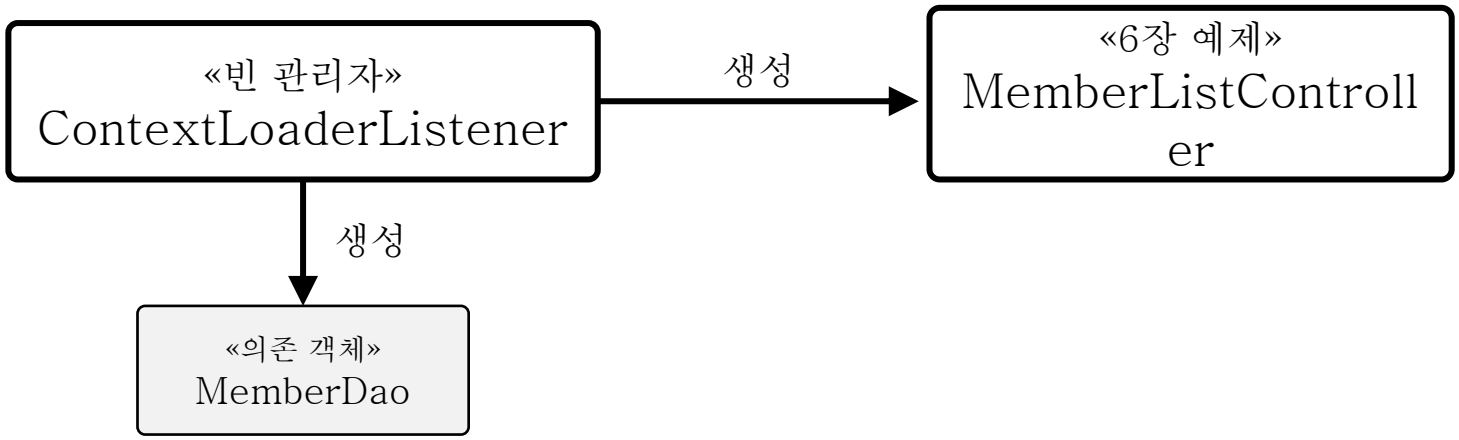
필요한 의존 객체를 사용 전에 미리 주입해 두기




```
public class ContextLoaderListener implements ServletContextListener {
  public void contextInitialized(ServletContextEvent event) {
    …
    MemberDao memberDao = new MemberDao();
    …
    sc.setAttribute("/member/list.do",
        new MemberListController().setMemberDao(memberDao));
```

6.3 DI를 이용한 빈 의존성 관리

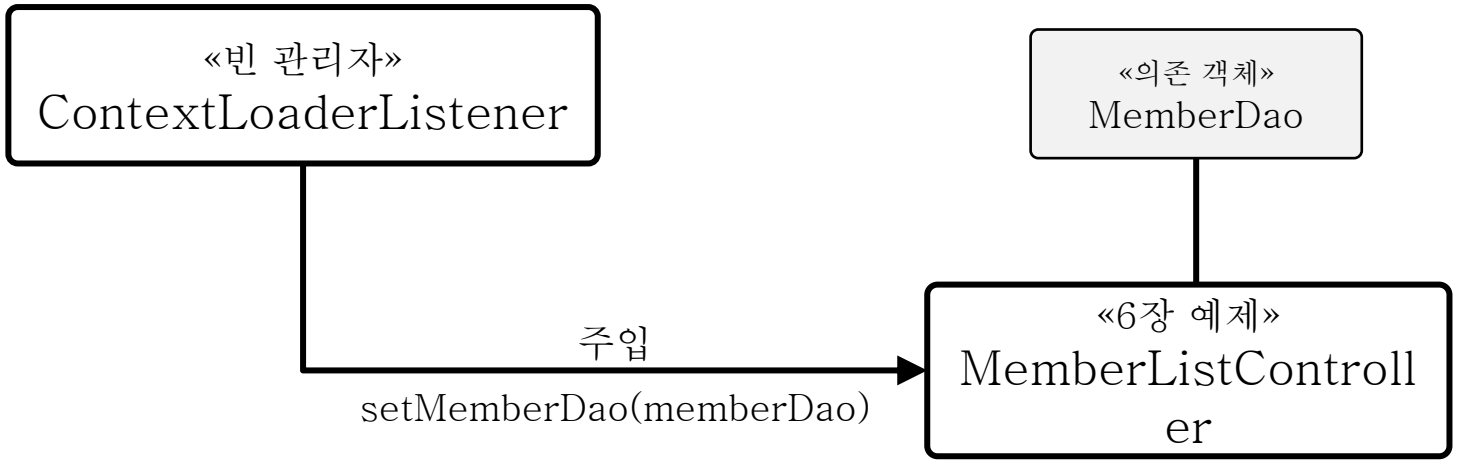
필요한 의존 객체를 사용 전에 미리 주입해 두기




```
public class ContextLoaderListener implements ServletContextListener {
  public void contextInitialized(ServletContextEvent event) {
    …
    MemberDao memberDao = new MemberDao();
    …
    sc.setAttribute("/member/list.do",
        new MemberListController().setMemberDao(memberDao));
```

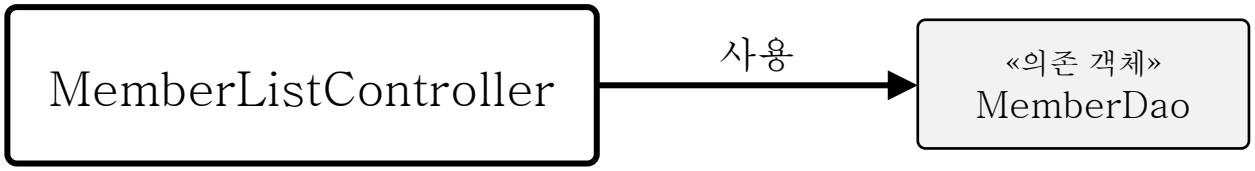
6.3 DI를 이용한 빈 의존성 관리

필요한 의존 객체를 사용 전에 미리 주입해 두기




```
public class ContextLoaderListener implements ServletContextListener {
  public void contextInitialized(ServletContextEvent event) {
    …
    MemberDao memberDao = new MemberDao();
    …
    sc.setAttribute("/member/list.do",
         new MemberListController().setMemberDao(memberDao));
```

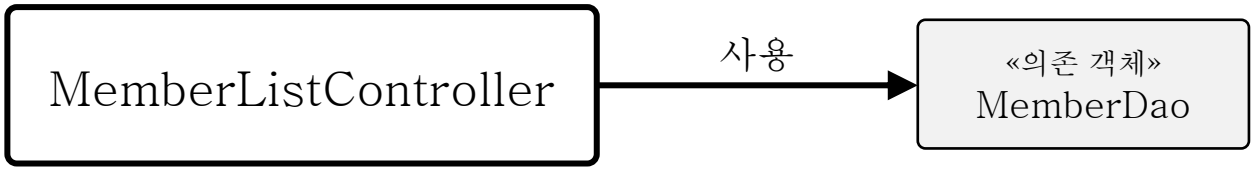
의존 객체 주입을 위한 코드 준비




```
public class MemberListController implements Controller {
  MemberDao memberDao;

  public MemberListController setMemberDao(MemberDao memberDao) {
    this.memberDao = memberDao;
    return this;
  }
  public String execute(Map<String, Object> model) throws Exception {
    model.put("members", memberDao.selectList());
    return "/member/MemberList.jsp";
  }
}
```

6.3 DI를 이용한 빈 의존성 관리

의존 객체 주입을 위한 코드 준비
MemberDao 인스턴스 변수와 셋터 메서드 추가



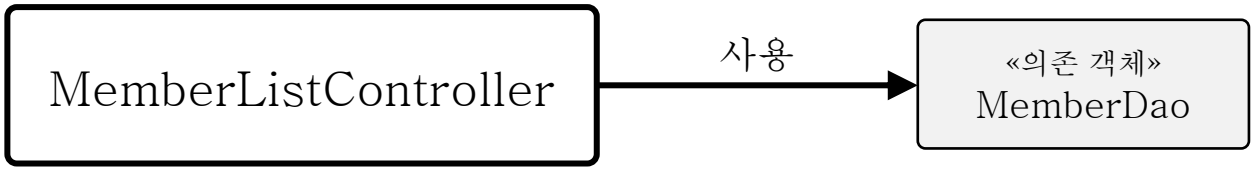

```
public class MemberListController implements Controller {
 MemberDao memberDao;

 public MemberListController setMemberDao(MemberDao memberDao) {
  this.memberDao = memberDao;
  return this;
 }
 public String execute(Map<String, Object> model) throws Exception {
  model.put("members", memberDao.selectList());
  return "/member/MemberList.jsp";
 }
}
```

의존 객체와 느슨한 연대

인터페이스를 이용하여 대체가 쉽게 하자!

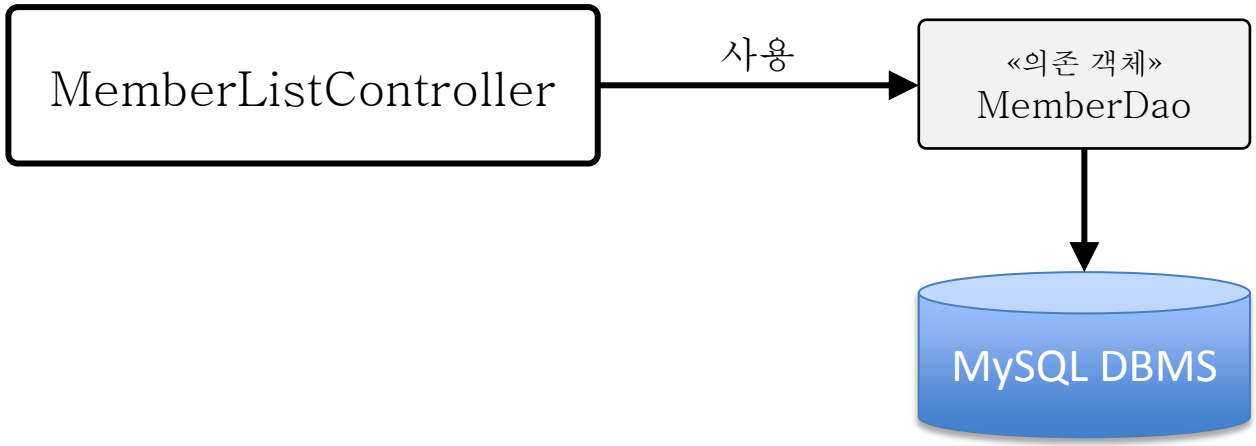
6.3 DI를 이용한 빈 의존성 관리

기존 방식: MemberDao가 클래스이다.



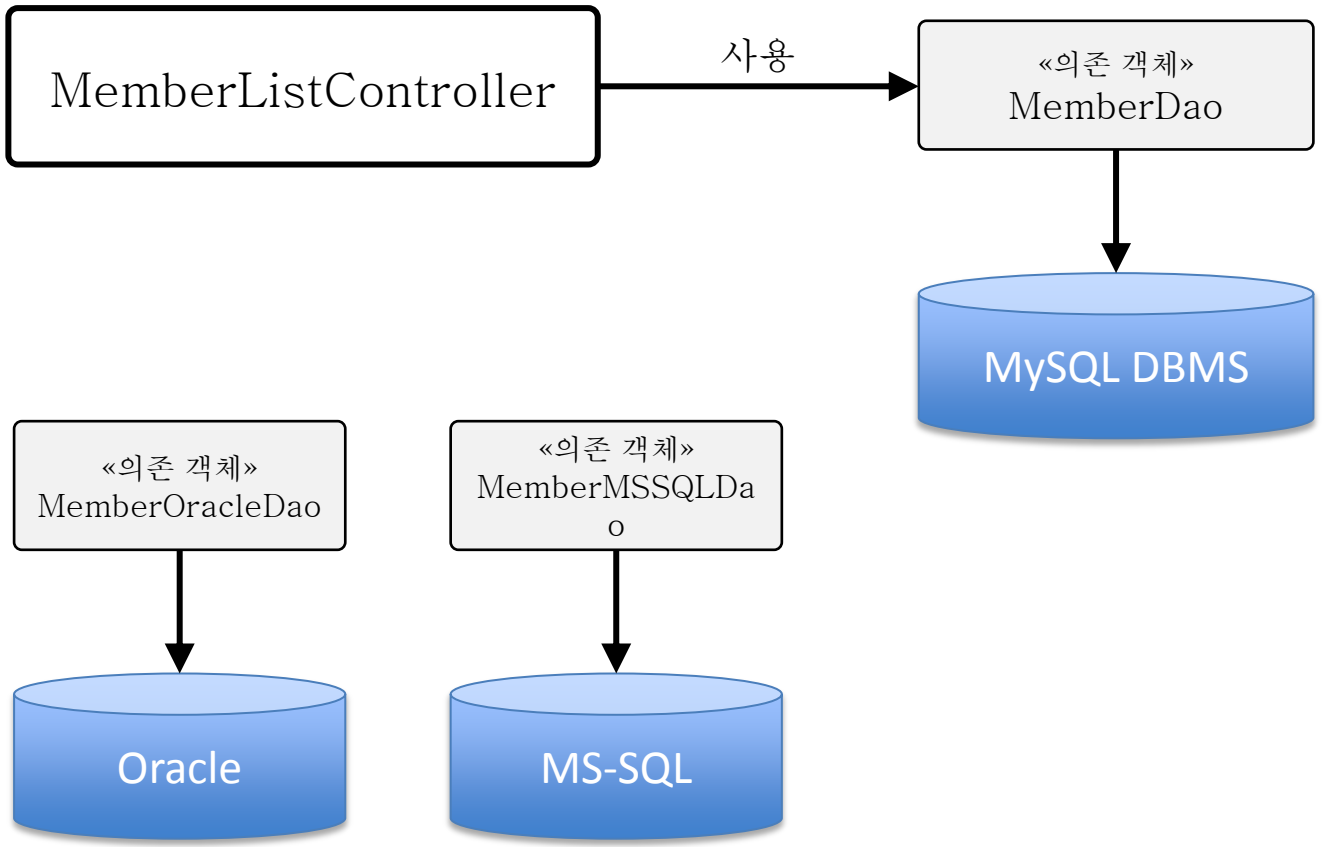

```
public class MemberDao {
  DataSource ds;

  public void setDataSource(DataSource ds) {
    this.ds = ds;
  }

  public List<Member> selectList() throws Exception { … }
  …
}
```

SQL문이 MySQL DBMS에 맞춰져 있다.



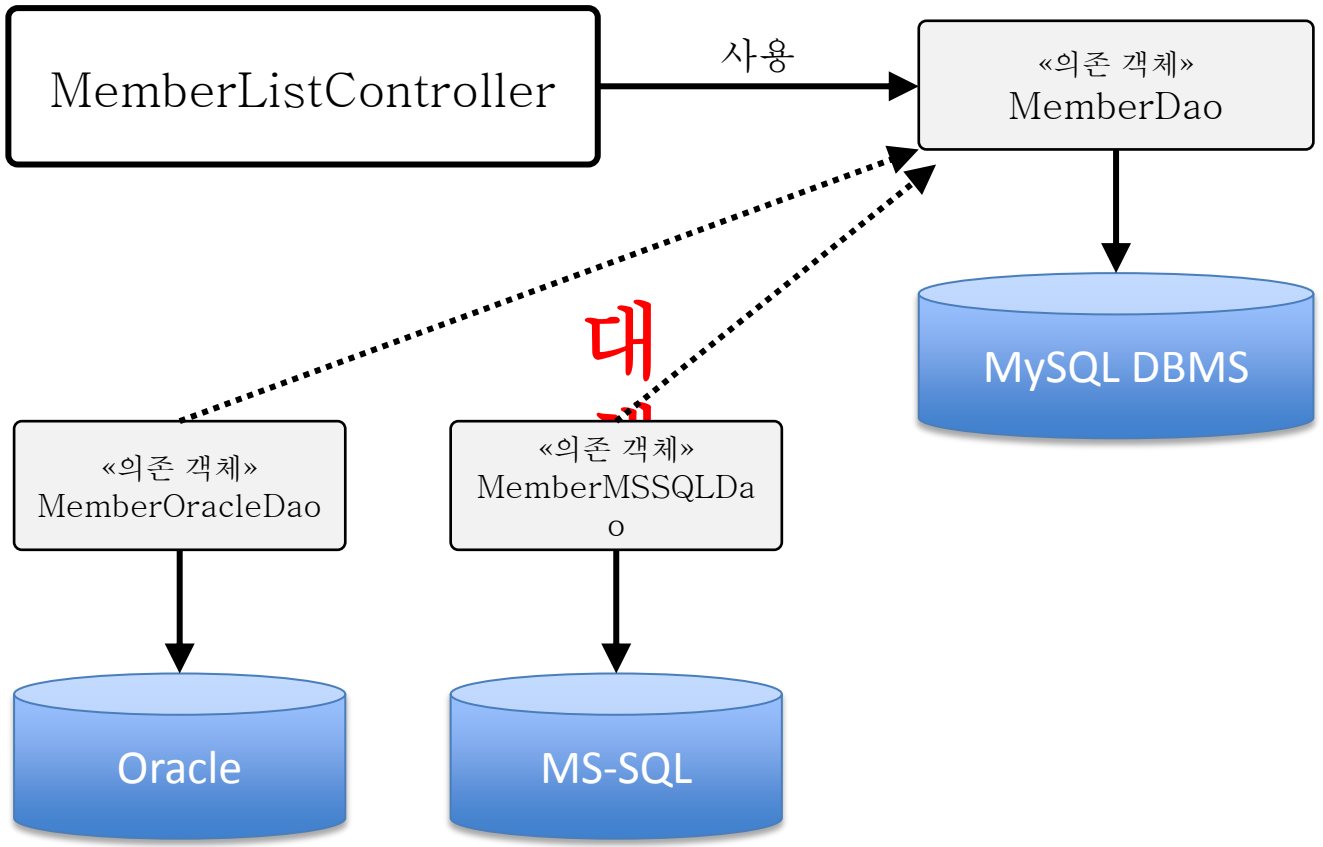
6.3 DI를 이용한 빈 의존성 관리

DBMS 마다 DAO를 따로 만든다면,

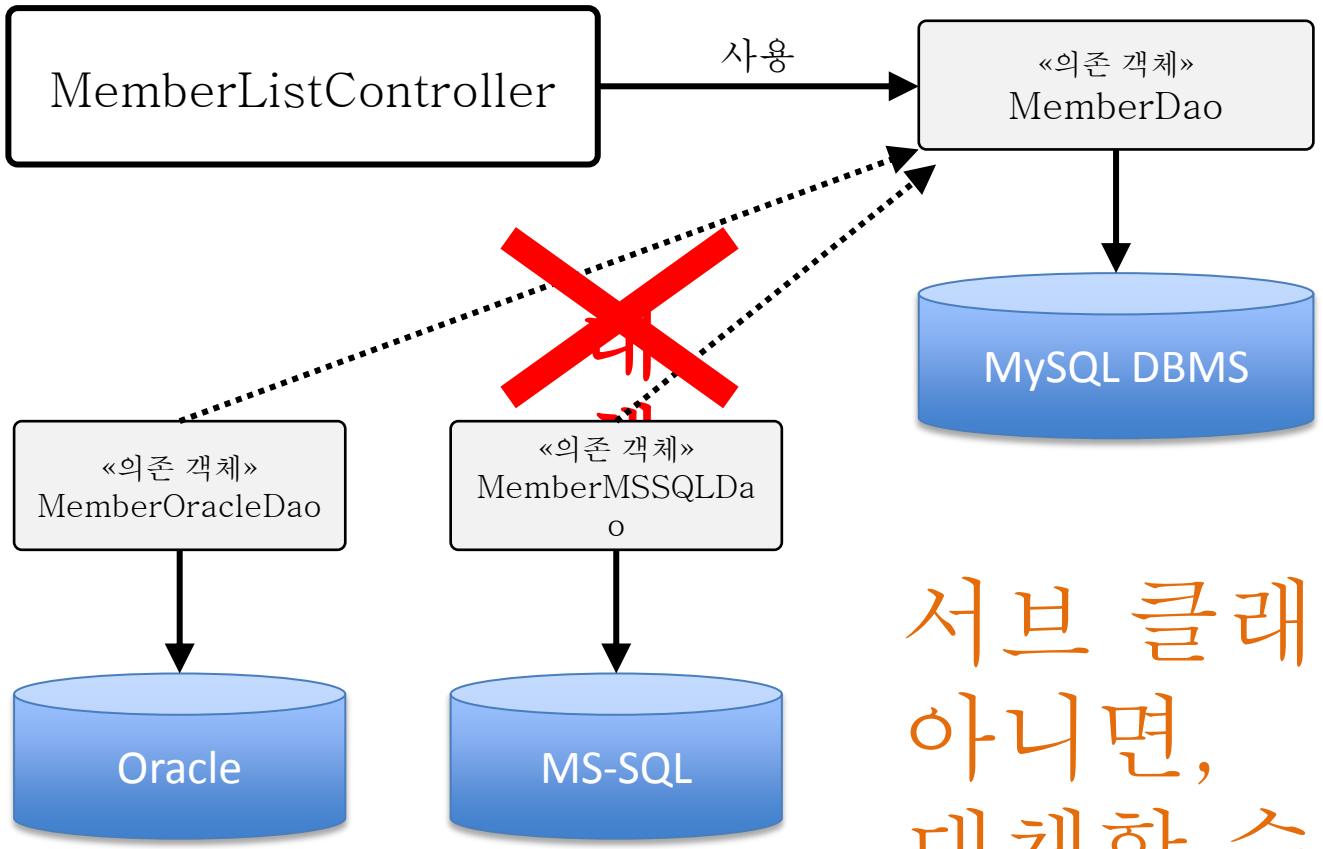


6.3 DI를 이용한 빈 의존성 관리

MemberDao 자리를 다른 DAO 클래스로 대체할 수 있는가?

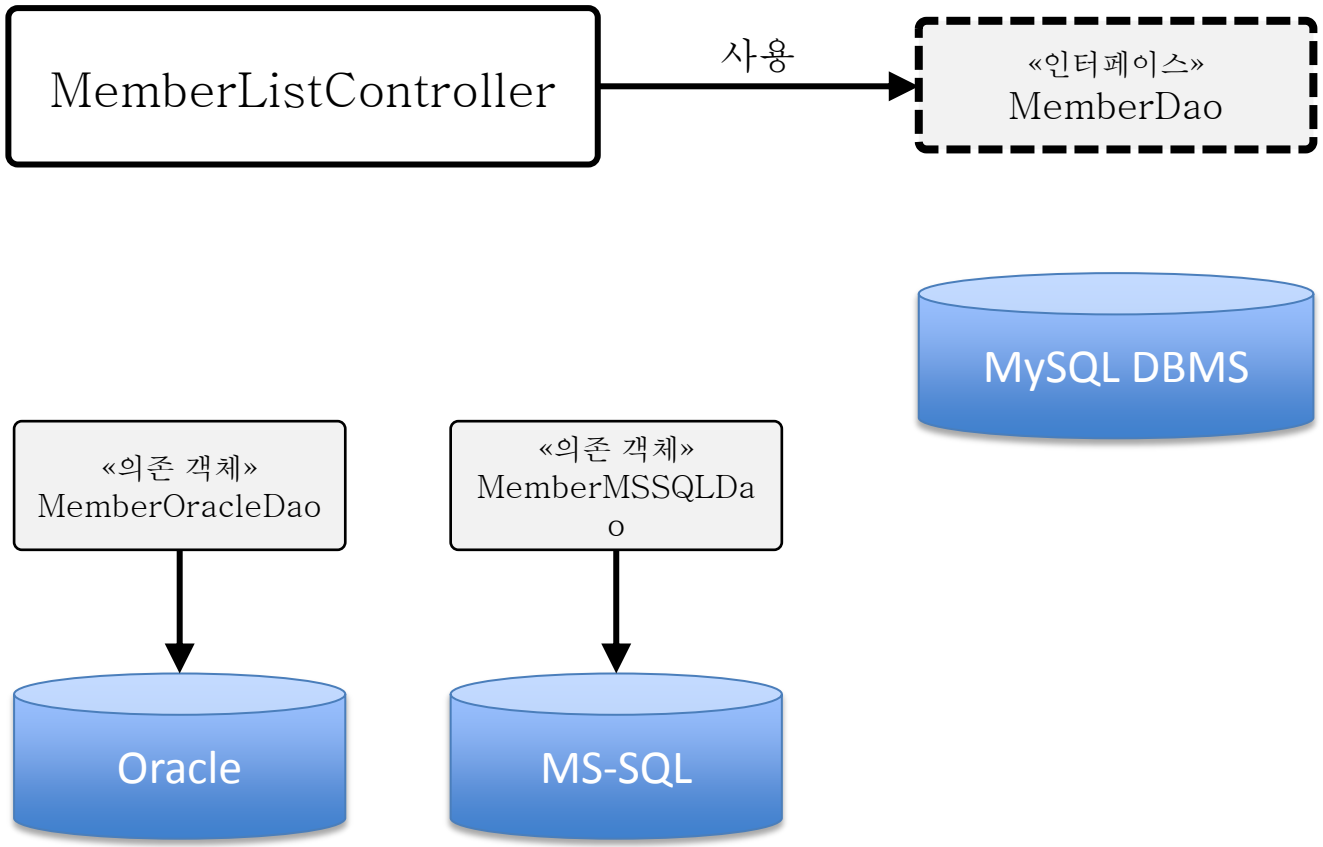


MemberDao 자리를 다른 DAO 클래스로 대체할 수 있는가?



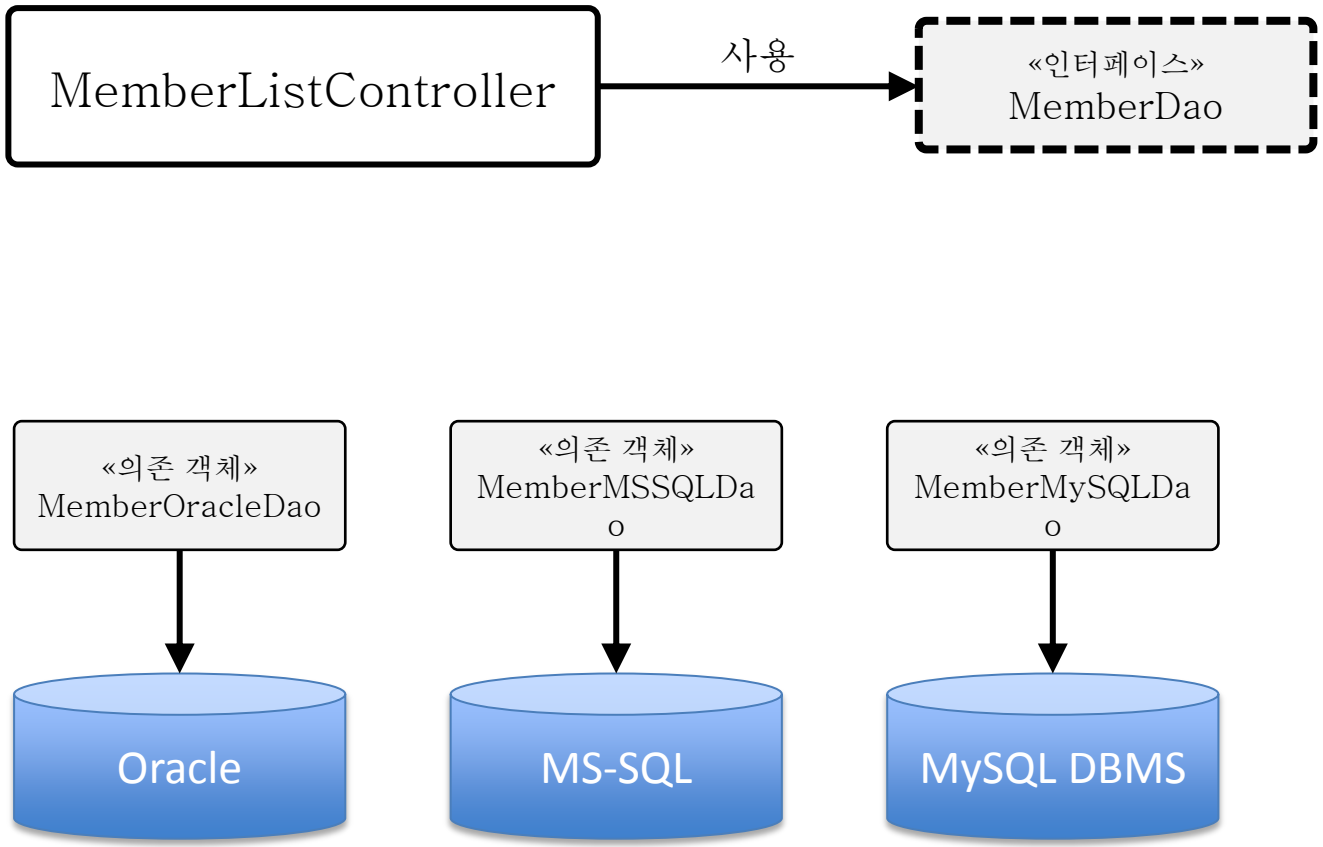
서브 클래스가 아니면, 대체할 수 없다!

해결책? MemberDao를 인터페이스로 선언



6.3 DI를 이용한 빈 의존성 관리

기존의 MemberDao는 MySQL 전용 DAO로 만든다.



6.3 DI를 이용한 빈 의존성 관리

모든 DAO 클래스는 MemberDao 인터페이스를구현한다.