6.4 리플랙션 API를 이용하여 프런트 컨트롤러 개선하기

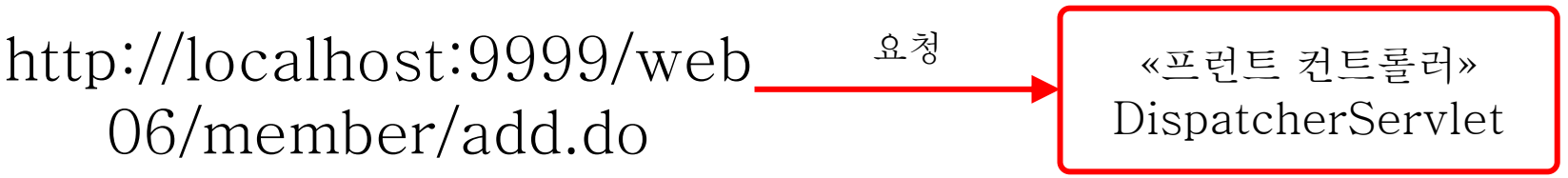
6.4 리플랙션 API를 이용하여 프런트 컨트롤러 개선하기

"학습할 내용"

파라미터 값을 페이지 컨트롤러에게 전달하는 방법을 자동화

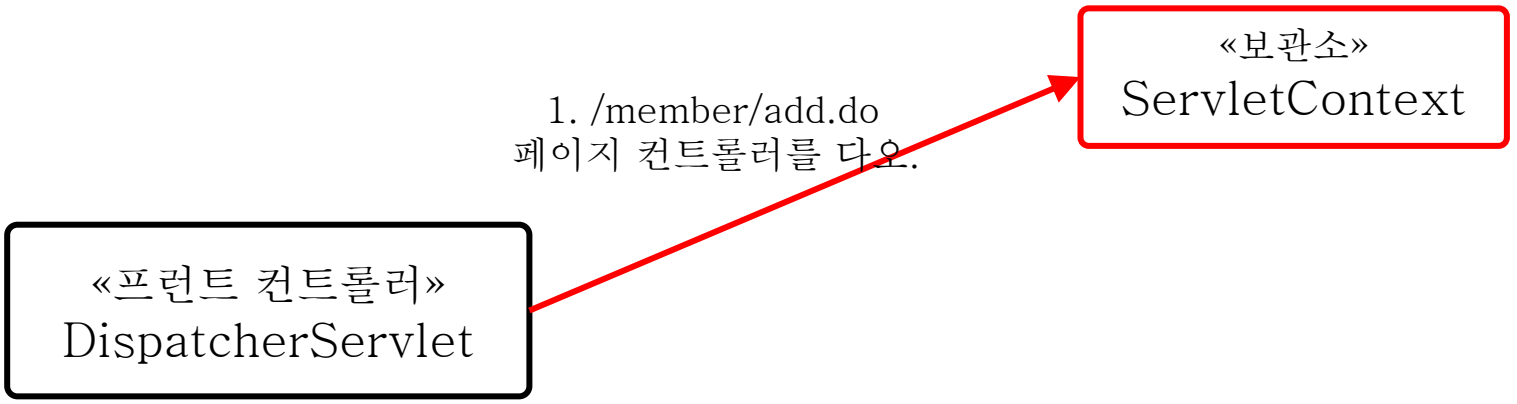
6.4 리플랙션 API를 이용하여 프런트 컨트롤러 개선하기

지금까지는…



6.4 리플랙션 API를 이용하여 프런트 컨트롤러 개선하기

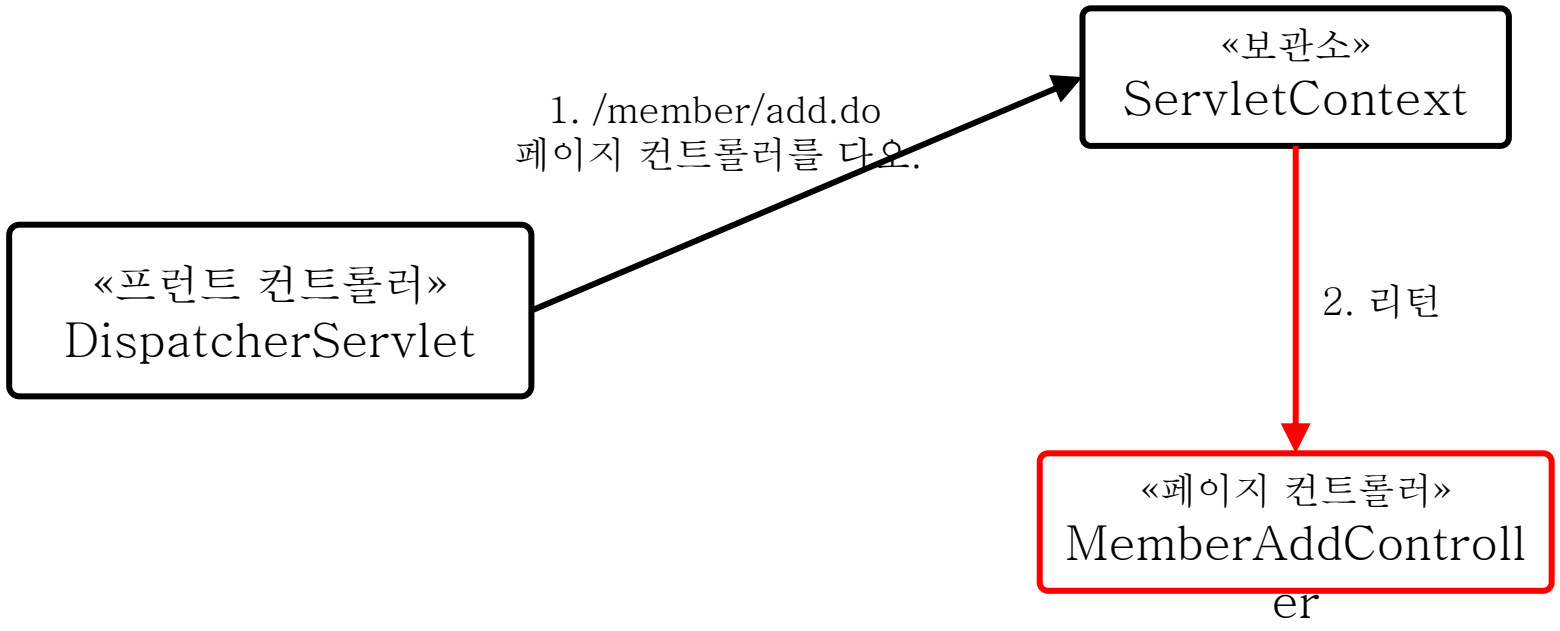
서블릿 컨텍스트 보관소에서 페이지 컨트롤러를 찾는다.




```
ServletContext sc = this.getServletContext();
…
Controller pageController = (Controller) sc.getAttribute(servletPath);
```

6.4 리플랙션 API를 이용하여 프런트 컨트롤러 개선하기

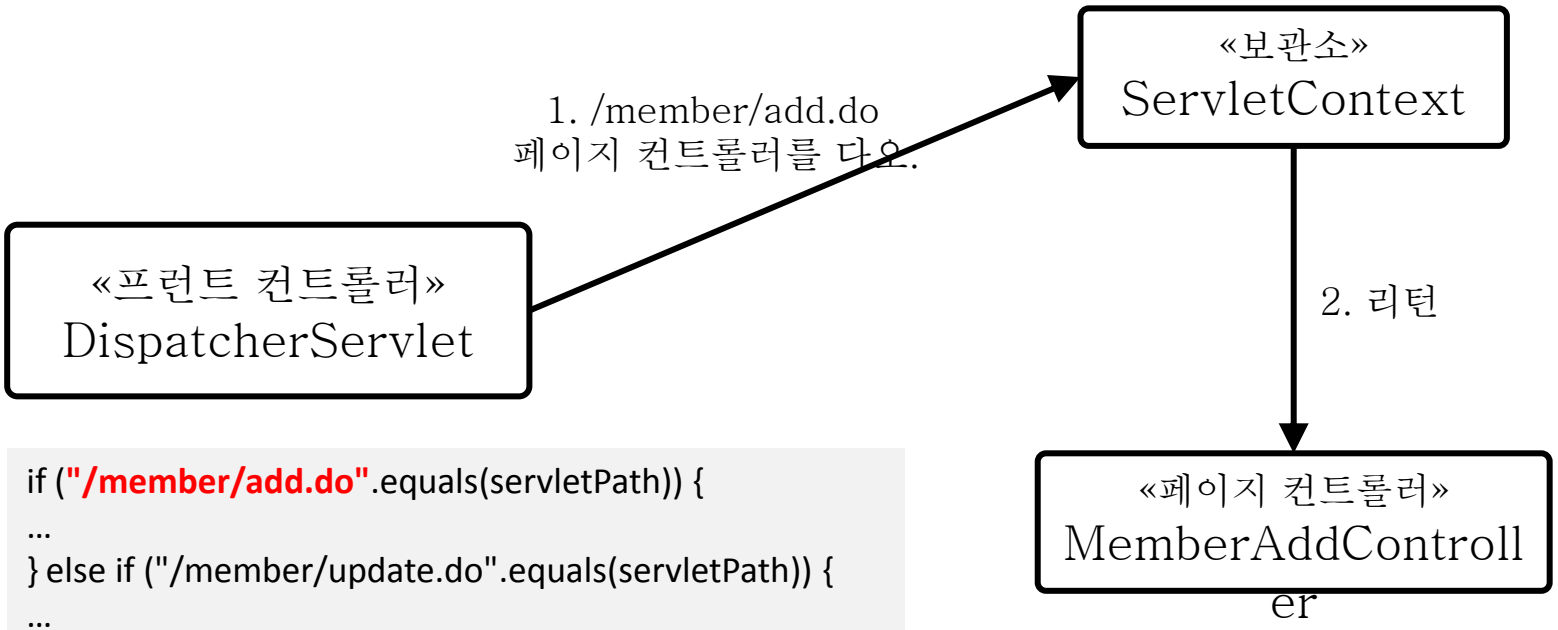
페이지 컨트롤러를 찾아 주면,




```
ServletContext sc = this.getServletContext();
…
Controller pageController = (Controller) sc.getAttribute(servletPath);
```

6.4 리플랙션 API를 이용하여 프런트 컨트롤러 개선하기

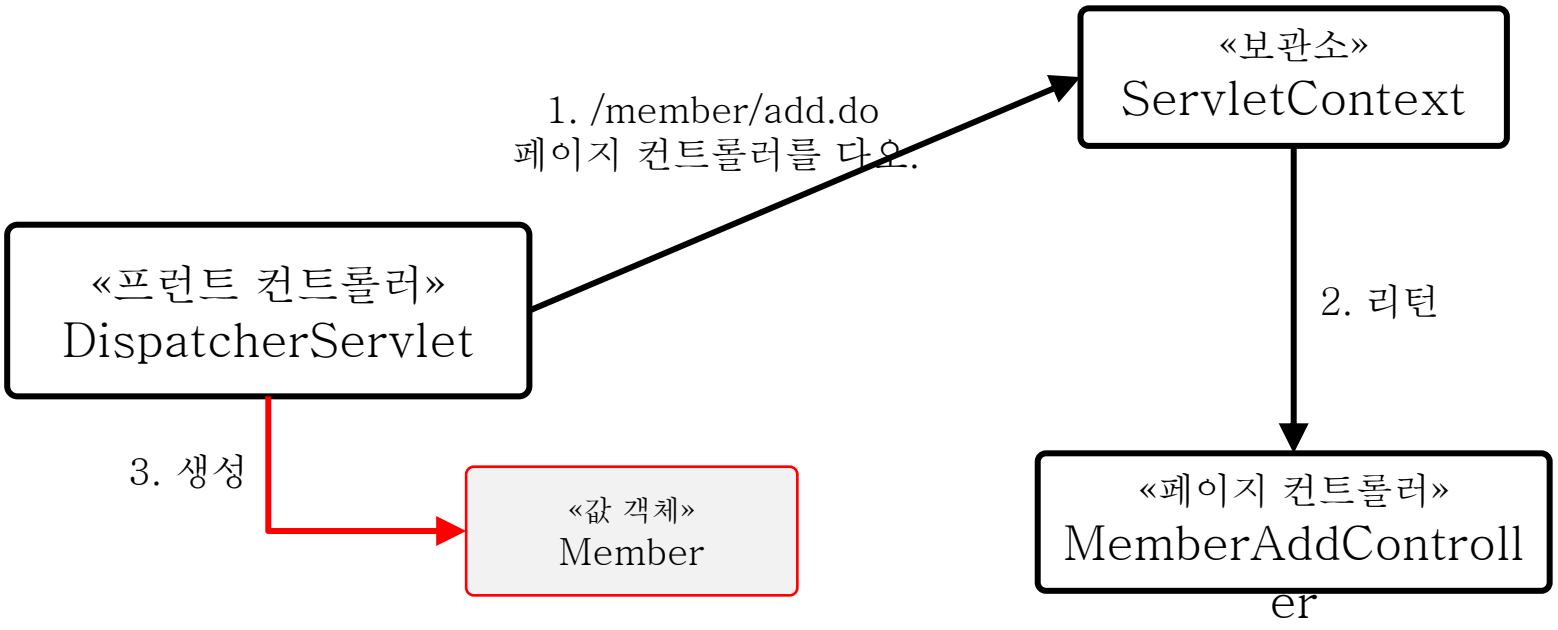
페이지 컨트롤러가 누구냐에 따라




```
if ("/member/add.do".equals(servletPath)) {
...
} else if ("/member/update.do".equals(servletPath)) {
...
} else if ("/member/delete.do".equals(servletPath)) {
...
} else if ("/auth/login.do".equals(servletPath)) {
...
}
```

6.4 리플랙션 API를 이용하여 프런트 컨트롤러 개선하기

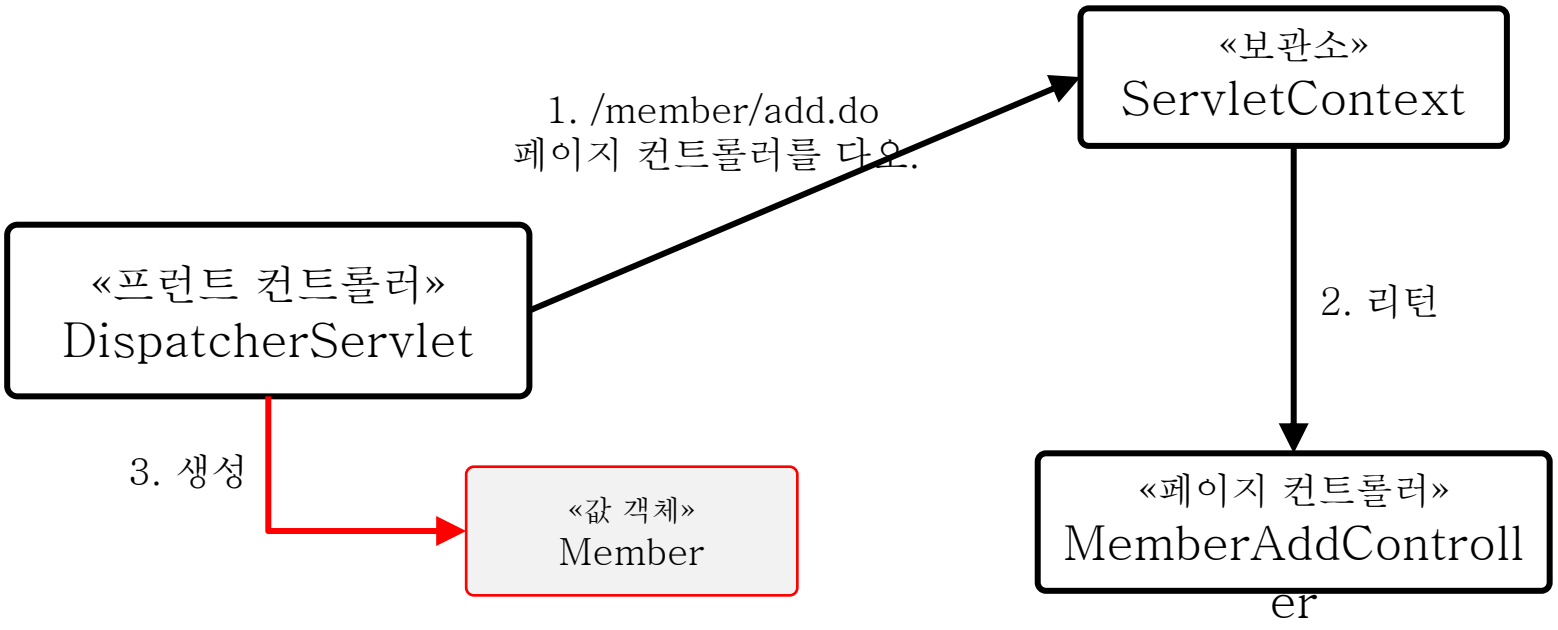
페이지 컨트롤러가 필요로 하는 값 객체를 준비한다.




```
new Member()
        .setEmail(request.getParameter("email"))
        .setPassword(request.getParameter("password"))
        .setName(request.getParameter("name"))
```

6.4 리플랙션 API를 이용하여 프런트 컨트롤러 개선하기

파라미터 값을 직접 꺼내 값 객체에 할당



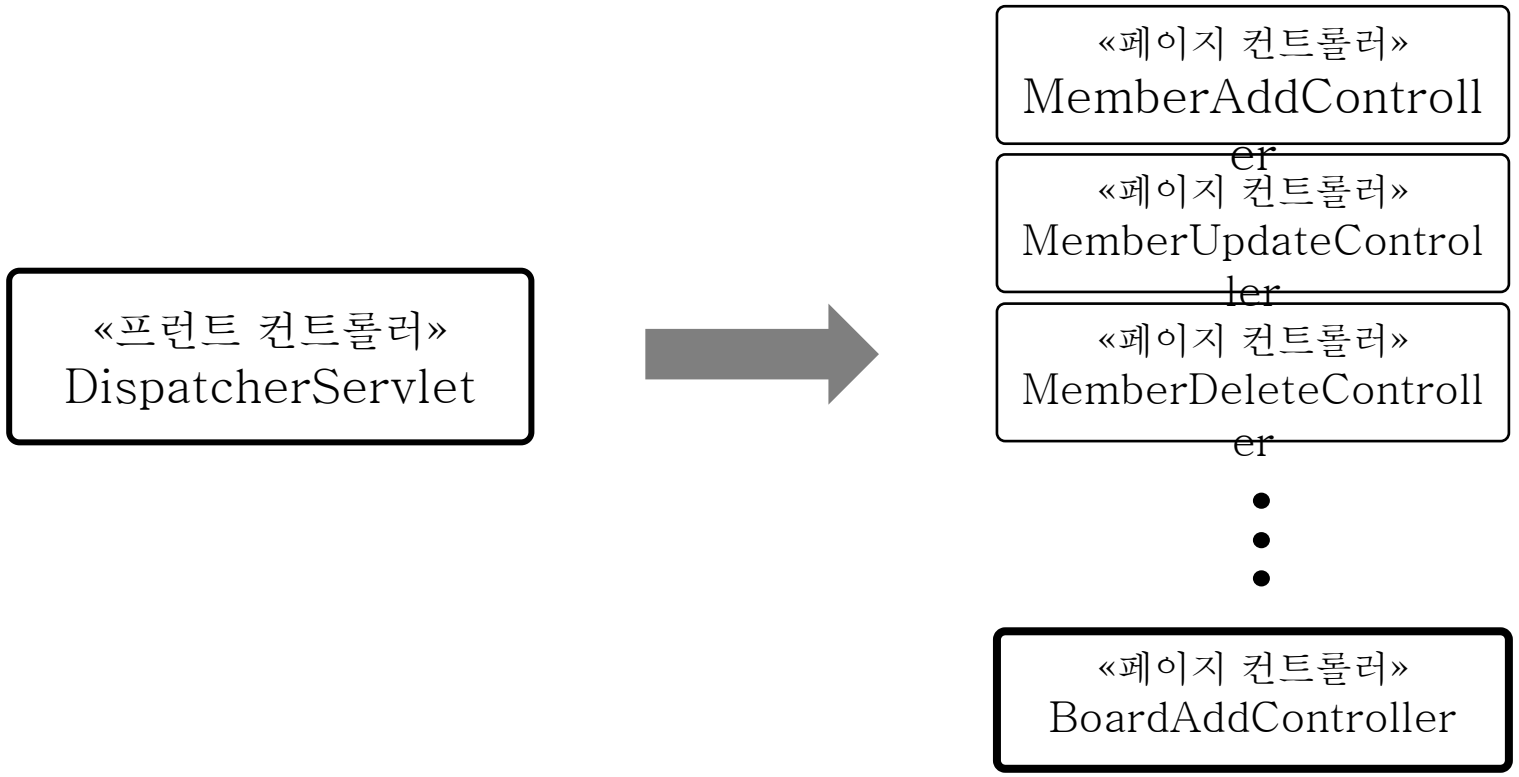

```
new Member()
        .setEmail(request.getParameter("email"))
        .setPassword(request.getParameter("password"))
        .setName(request.getParameter("name"))
```

6.4 리플랙션 API를 이용하여 프런트 컨트롤러 개선하기

기존 방식의 문제점

6.4 리플랙션 API를 이용하여 프런트 컨트롤러 개선하기

새 페이지 컨트롤러가 추가되면,



6.4 리플랙션 API를 이용하여 프런트 컨트롤러 개선하기

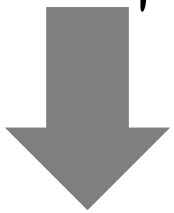
프런트 컨트롤러에 값 객체를 준비하는 조건문을 추가해야 한다.

```
…
} else if ("/member/update.do".equals(servletPath)) {
      if (request.getParameter("email") != null) {
        model.put("member", new Member()…);
      } else {
        model.put("no", new Integer(request.getParameter("no")));
      }
} else if ("/member/delete.do".equals(servletPath)) {
      model.put("no", new Integer(request.getParameter("no")));
} else if ("/auth/login.do".equals(servletPath)) {
      if (request.getParameter("email") != null) {
        model.put("loginInfo", new Member()…);
      }
} else if ("/board/add.do".equals(servletPath)) {
      if (request.getParameter(”title") != null) {
        model.put(”board", new Board()…);
      } else {
        model.put("no", new Integer(request.getParameter("no")));
      }
}
```

페이지 컨트롤러를 추가하더라도 프런트 컨트롤러를 변경하지 않으려면?

6.4 리플랙션 API를 이용하여 프런트 컨트롤러 개선하기

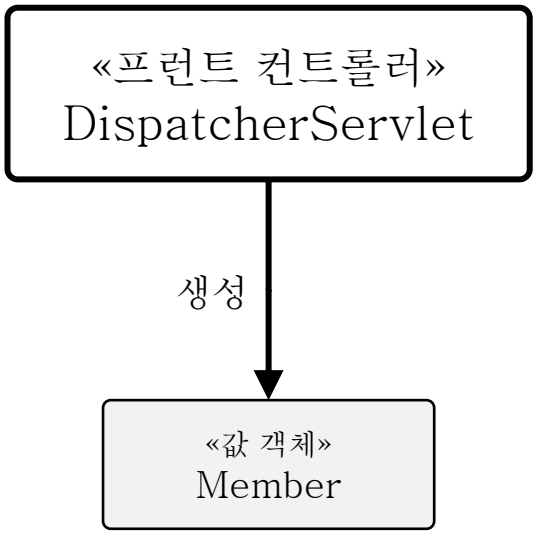
페이지 컨트롤러를
추가하더라도
프런트 컨트롤러를 변경하지
않으려면?



“구조 변경”

6.4 리플랙션 API를 이용하여 프런트 컨트롤러 개선하기

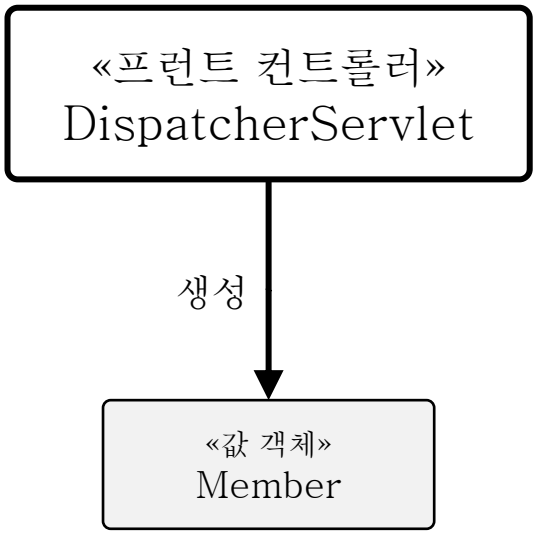
값 객체를 직접 생성하는 생성하는 코드를 제거!




```
if ("/member/add.do".equals(servletPath)) {
  if (request.getParameter("email") != null) {
    model.put("member", new Member()
        .setEmail(request.getParameter("email"))
        .setPassword(request.getParameter("password"))
        .setName(request.getParameter("name")));
  }
} else if ("/member/update.do".equals(servletPath)) {
  if (request.getParameter("email") != null) {
    model.put("member", new Member()
        .setNo(Integer.parseInt(request.getParameter("no")))
        .setEmail(request.getParameter("email"))
        .setName(request.getParameter("name")));
  } else {
    model.put("no",
                new Integer(request.getParameter("no")));
  }
}
...
```

6.4 리플랙션 API를 이용하여 프런트 컨트롤러 개선하기

값 객체를 직접 생성하는 생성하는 코드를 제거!



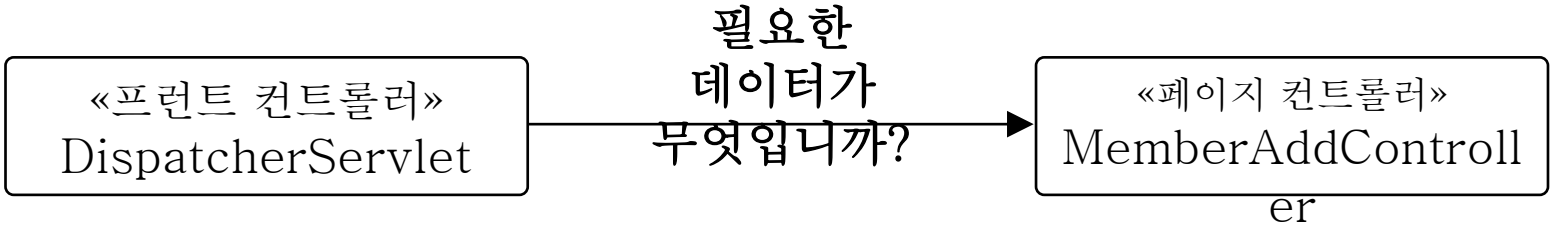

```
if ("/member/add.do".equals(servletPath)) {
  if (req      getParameter("email") != null) {
    mo         member", new Member()
        .set       uest.getParameter("e
        .setPas    request.getParam      sword"))
        .setName   getParame      e")));
  }
} else if ("/member/      (servletPath)) {
  if (request.getParamet          != null) {
    model.put("membe          ber()
        .setNo(Integer          tParameter("no")))
        .setEmail(re      Param      il"))
        .setNam      getParamet      ")));
  } else {
    mod      ,
          ew Integer(request.getParam      )));
  }
}
...
```

6.4 리플랙션 API를 이용하여 프런트 컨트롤러 개선하기

페이지 컨트롤러에게 필요한 데이터가 무엇인지 물어보고 그에 맞추어 준비해 준다.

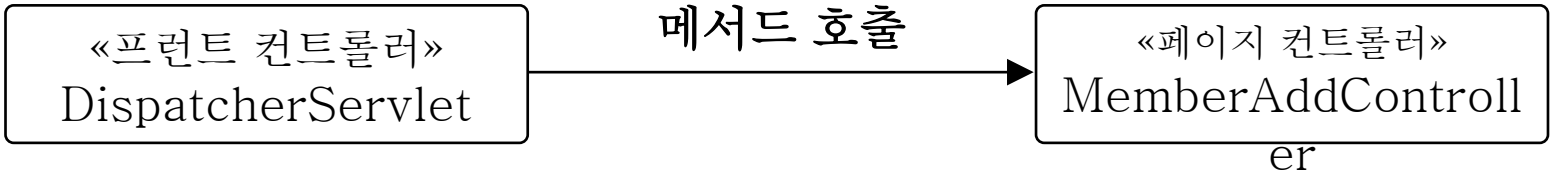
6.4 리플랙션 API를 이용하여 프런트 컨트롤러 개선하기

페이지 컨트롤러에게 필요한 데이터가 무엇인지 물어보고 그에 맞추어 준비해 준다.



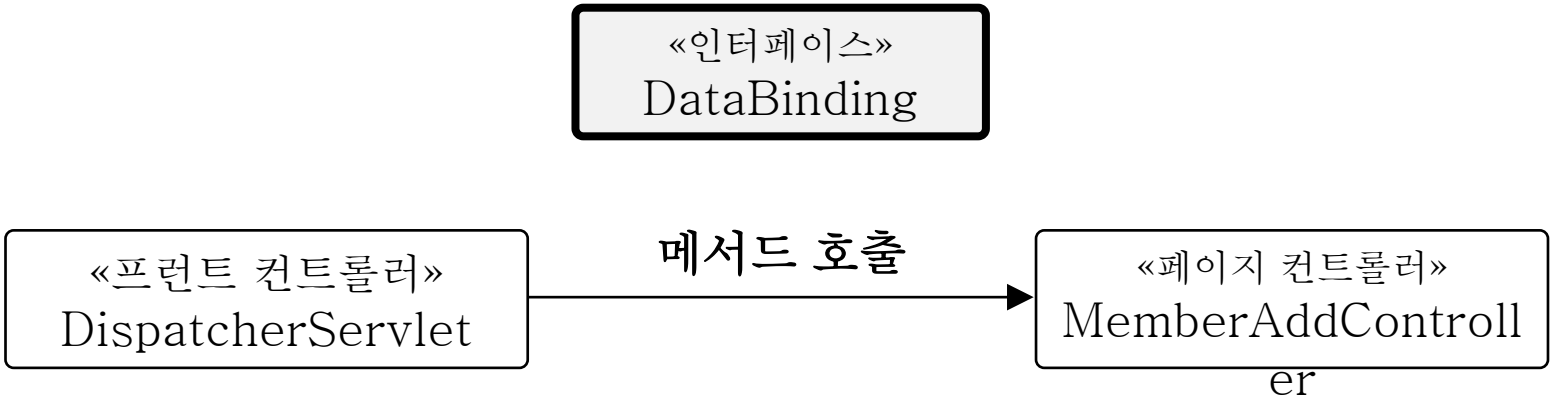
6.4 리플랙션 API를 이용하여 프런트 컨트롤러 개선하기

페이지 컨트롤러에게 필요한 데이터가 무엇인지 물어보고 그에 맞추어 준비해 준다.



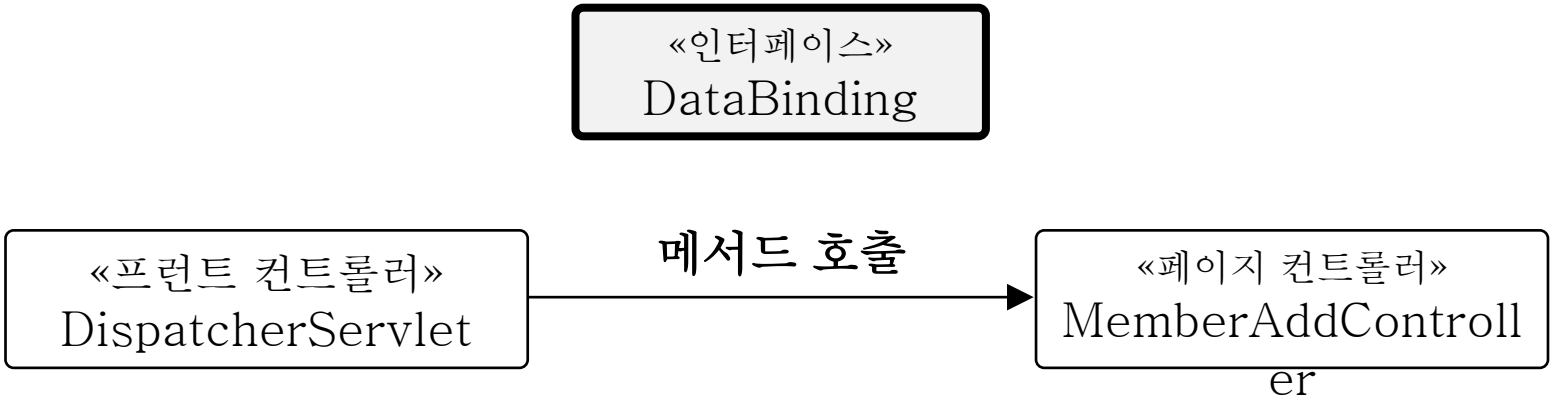
6.4 리플랙션 API를 이용하여 프런트 컨트롤러 개선하기

프런트 컨트롤러와 페이지 컨트롤러 사이에 호출 규칙 정의



6.4 리플랙션 API를 이용하여 프런트 컨트롤러 개선하기

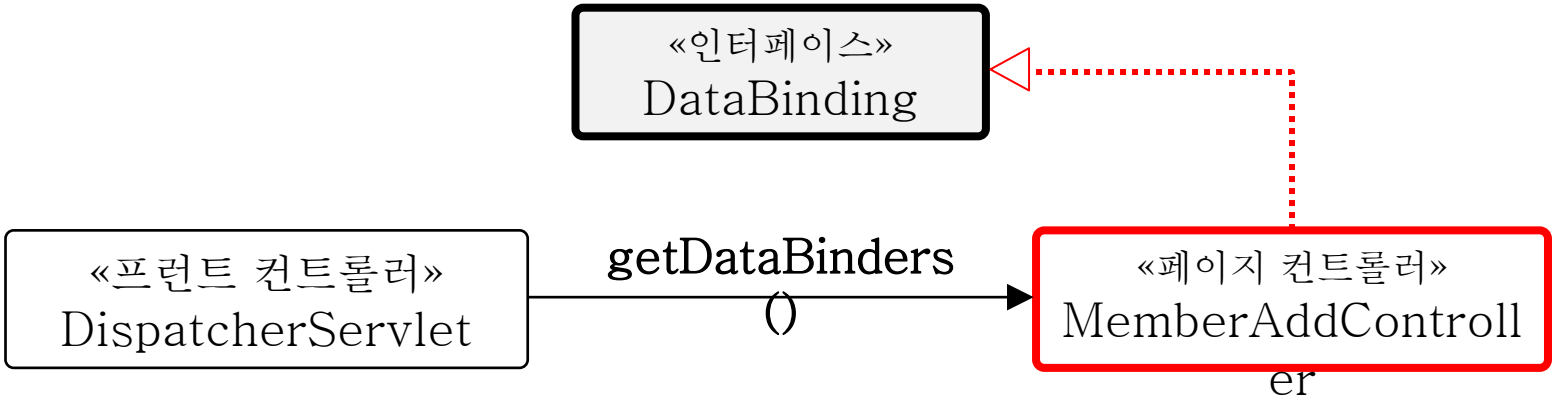
프런트 컨트롤러와 페이지 컨트롤러 사이에 호출 규칙 정의




```
public interface DataBinding {
  Object[] getDataBinders();
}
```

6.4 리플랙션 API를 이용하여 프런트 컨트롤러 개선하기

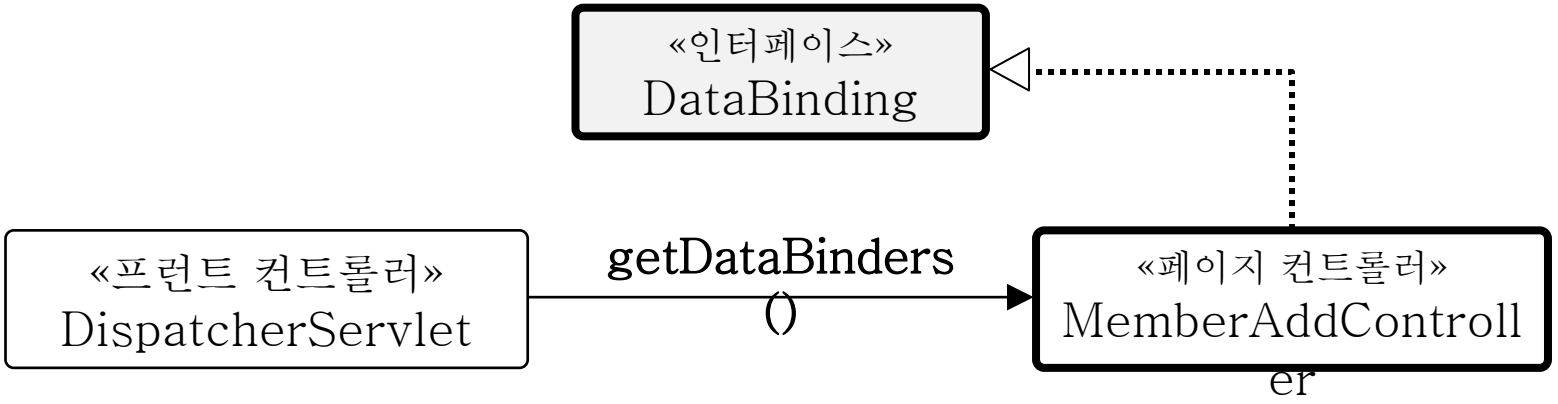
프런트 컨트롤러와 페이지 컨트롤러 사이에 호출 규칙 정의




```
public class MemberAddController implements Controller, DataBinding {
  …
  public Object[] getDataBinders() {
    return new Object[]{"member", spms.vo.Member.class};
  }
```

6.4 리플랙션 API를 이용하여 프런트 컨트롤러 개선하기

프런트 컨트롤러와 페이지 컨트롤러 사이에 호출 규칙 정의

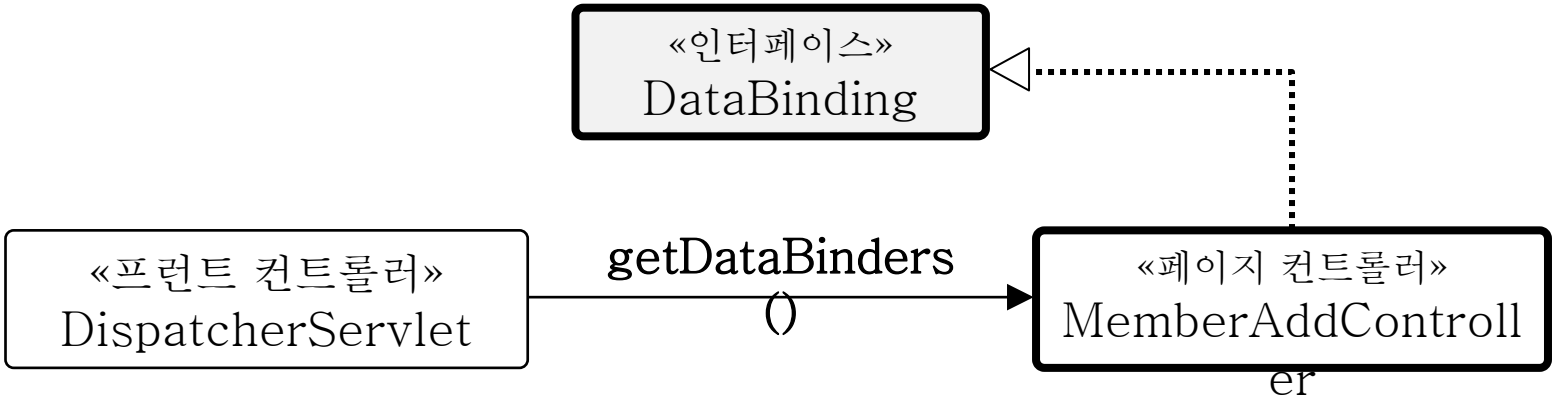



```
public class MemberAddController implements Controller, DataBinding {
 …
 public Object[] getDataBinders() {
  return new Object[]{"member", spms.vo.Member.class};
 }
```

값 객체 이름

6.4 리플랙션 API를 이용하여 프런트 컨트롤러 개선하기

프런트 컨트롤러와 페이지 컨트롤러 사이에 호출 규칙 정의




```
public class MemberAddController implements Controller, DataBinding {
  ...
  public Object[] getDataBinders() {
    return new Object[]{"member", spms.vo.Member.class};
  }
```

값 객체 이름

값 객체 클래스 정보

6.4 리플랙션 API를 이용하여 프린트 컨트롤러 개선하기

프린트 컨트롤러 변경

6.4 리플랙션 API를 이용하여 프런트 컨트롤러 개선하기

DataBinding을 구현한 페이지 컨트롤러에 대해서만 값 객체를 만들 준다.

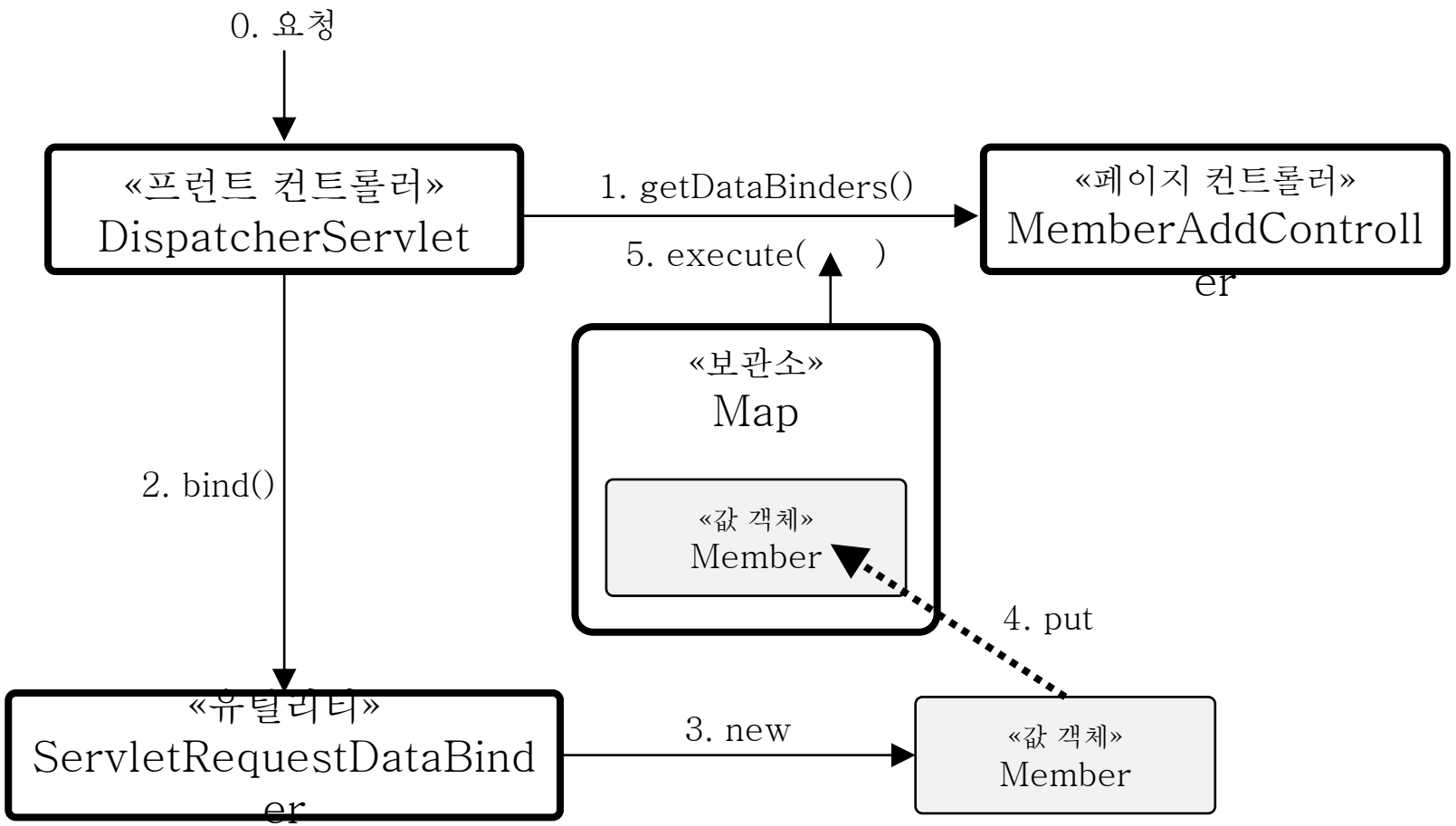
```
ServletContext sc = this.getServletContext();
HashMap<String,Object> model = new HashMap<String,Object>();
model.put("session", request.getSession());

Controller pageController = (Controller) sc.getAttribute(servletPath);

if (pageController instanceof DataBinding) {
  prepareRequestData(request, model, (DataBinding)pageController);
}
```

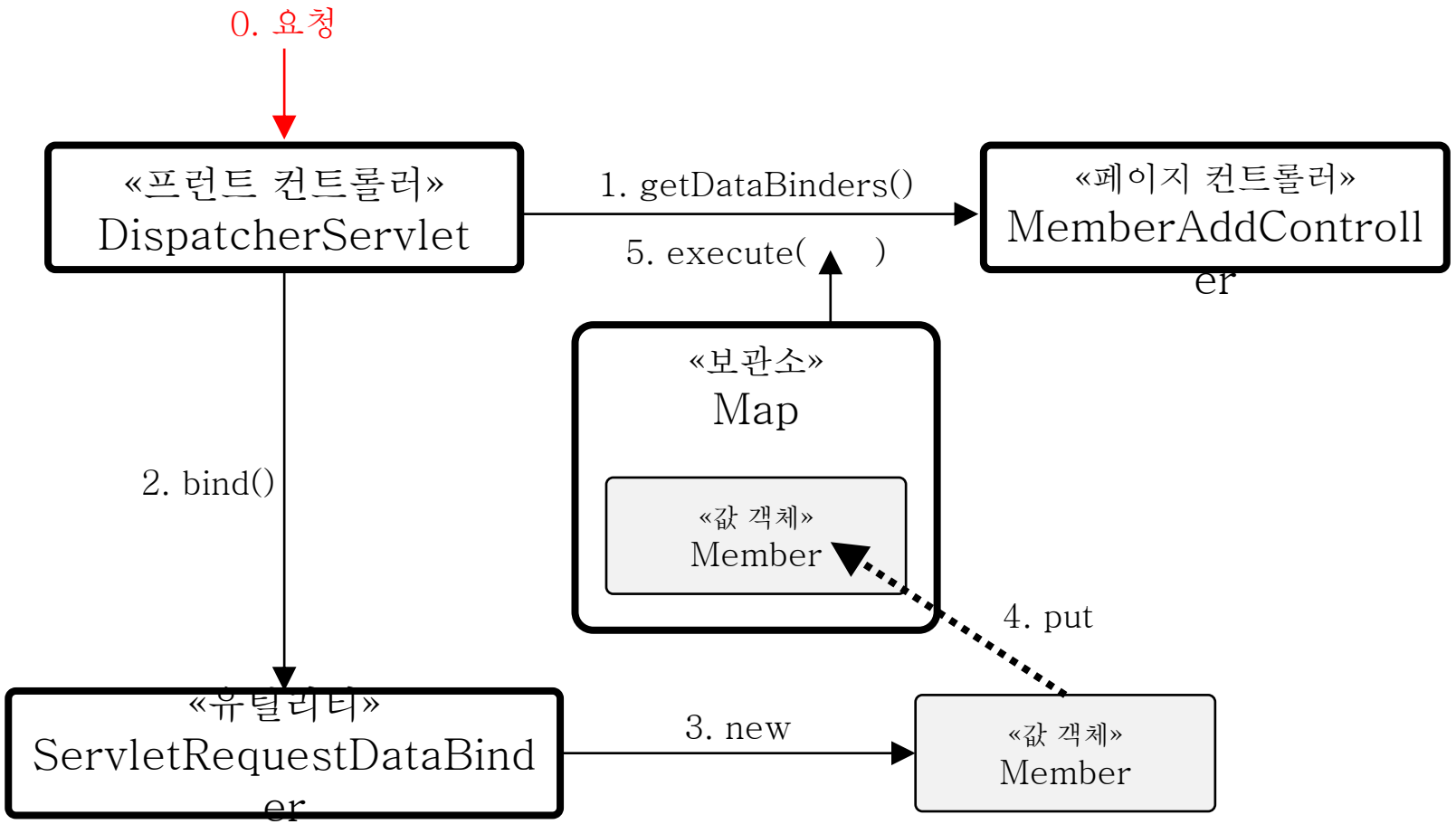
6.4 리플랙션 API를 이용하여 프런트 컨트롤러 개선하기

값 객체를 준비하는 과정



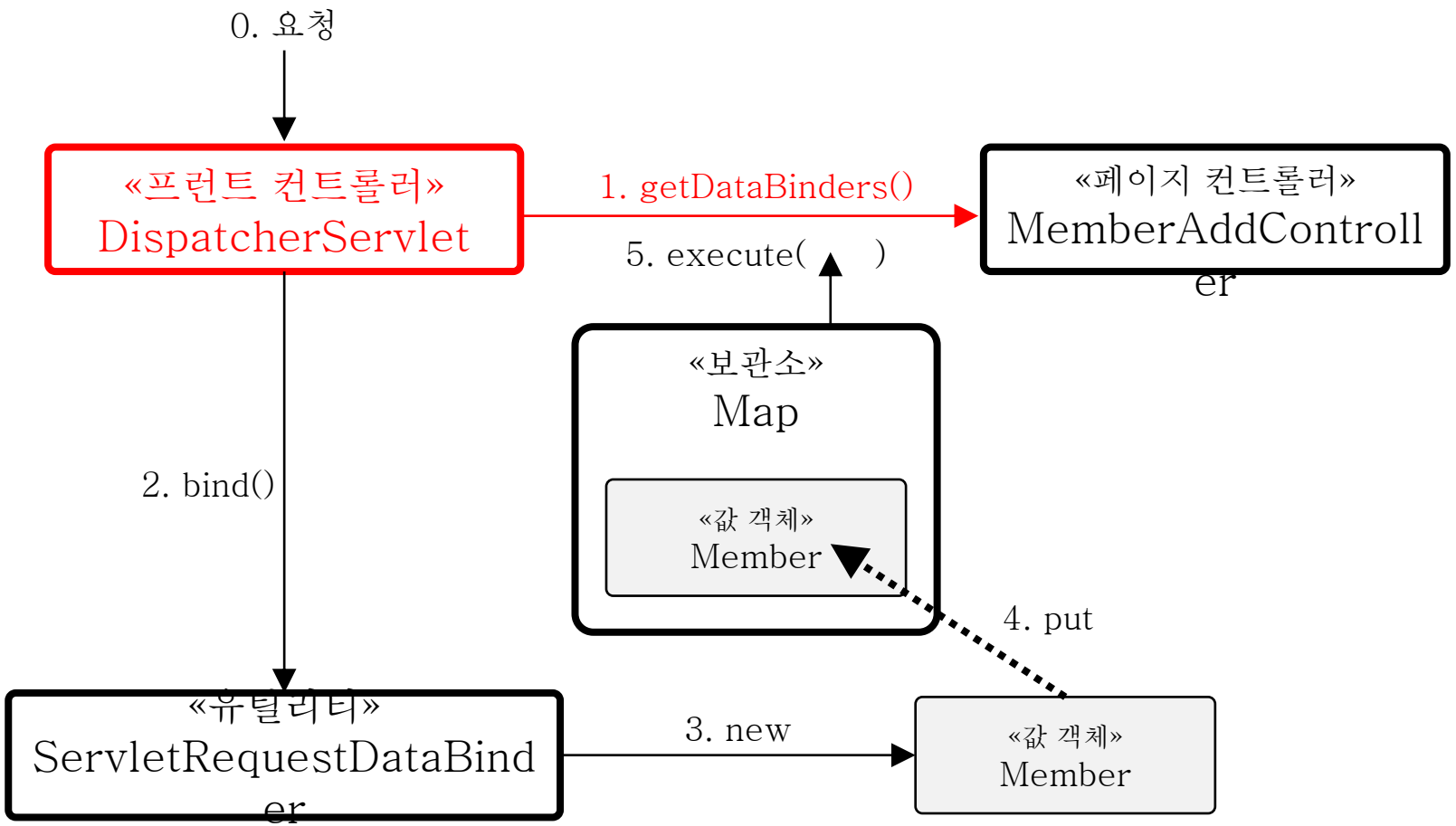
6.4 리플랙션 API를 이용하여 프런트 컨트롤러 개선하기

값 객체를 준비하는 과정



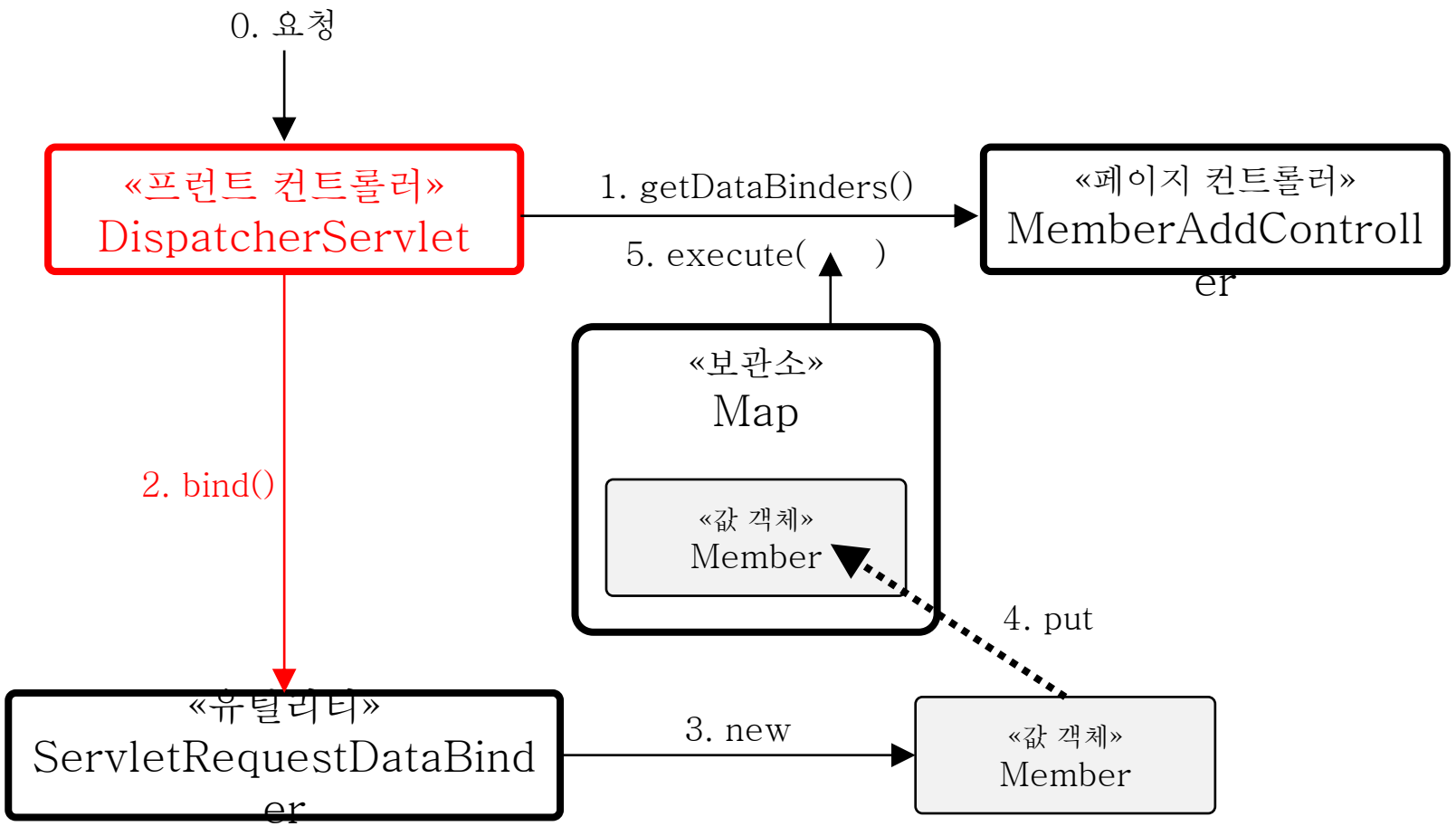
6.4 리플랙션 API를 이용하여 프런트 컨트롤러 개선하기

값 객체를 준비하는 과정



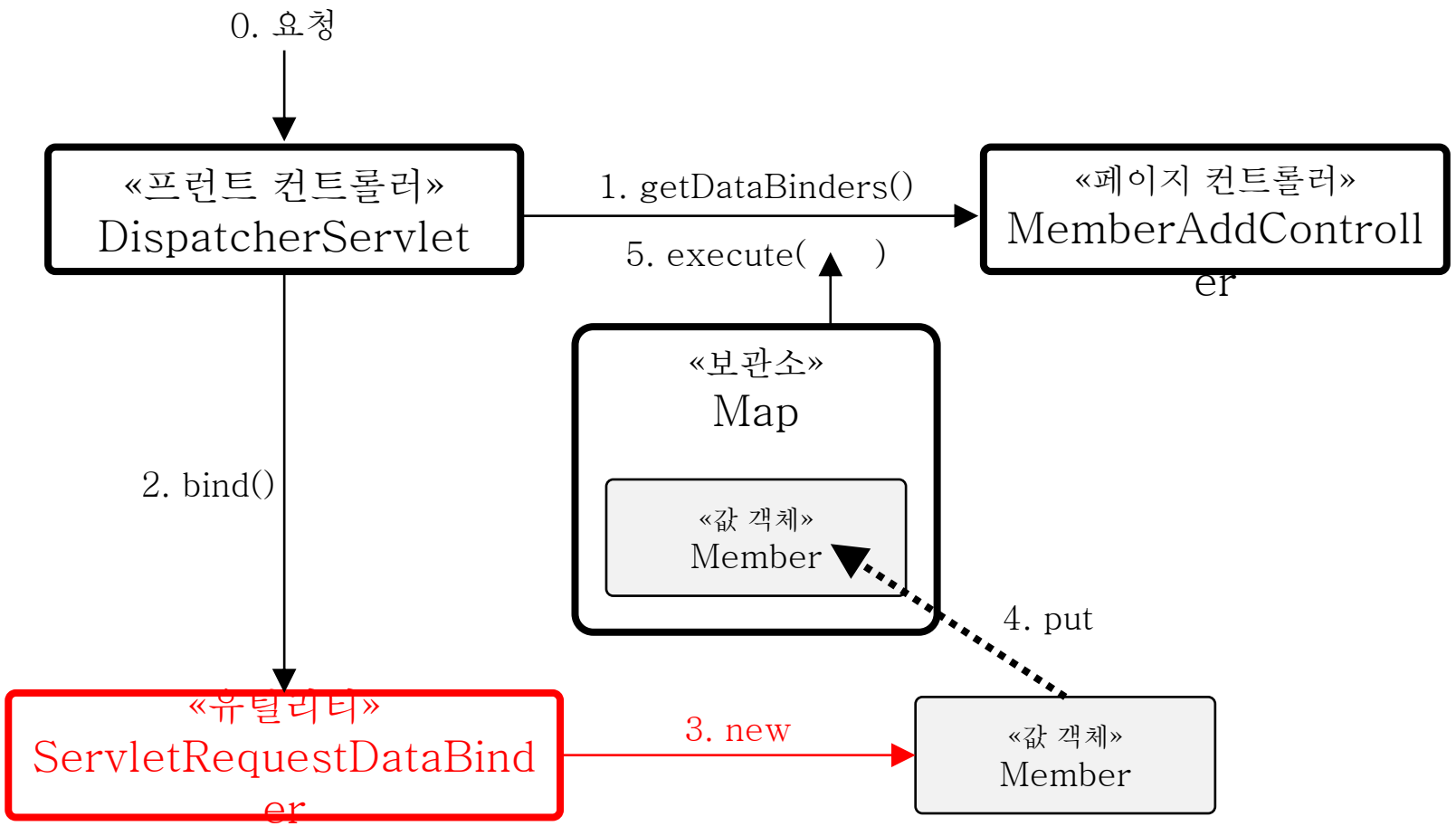
6.4 리플랙션 API를 이용하여 프런트 컨트롤러 개선하기

값 객체를 준비하는 과정



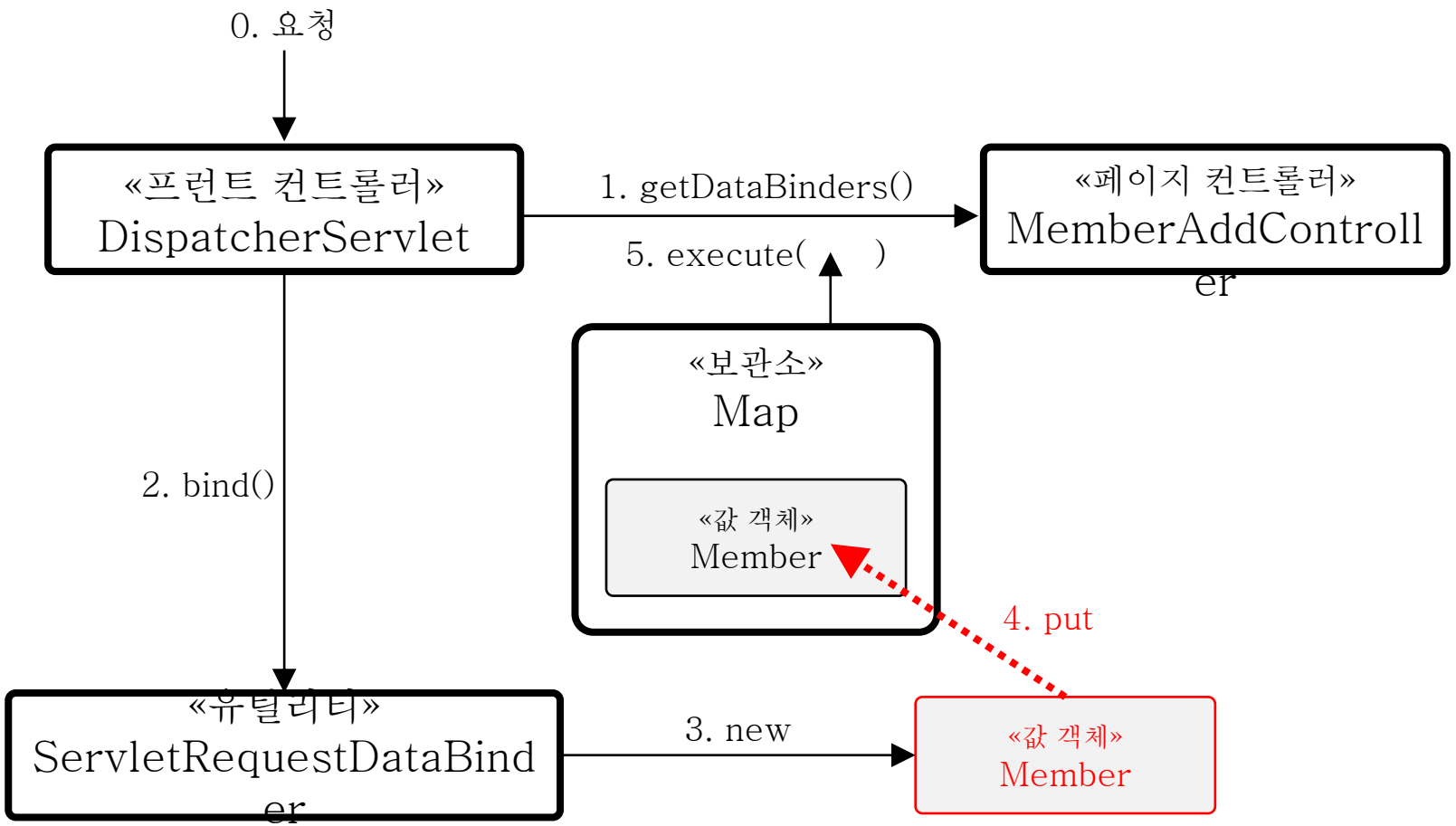
6.4 리플랙션 API를 이용하여 프런트 컨트롤러 개선하기

값 객체를 준비하는 과정



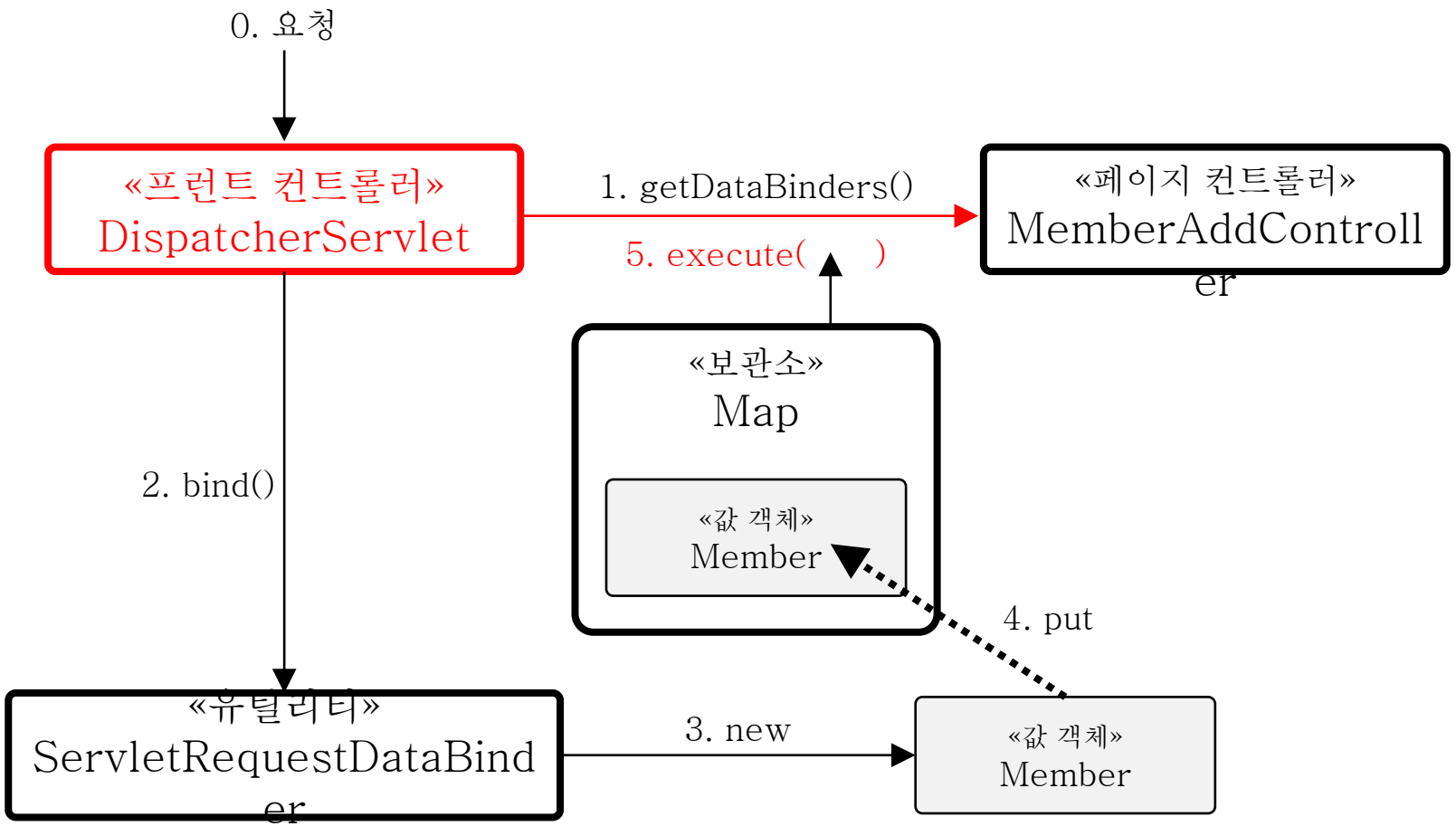
6.4 리플랙션 API를 이용하여 프런트 컨트롤러 개선하기

값 객체를 준비하는 과정



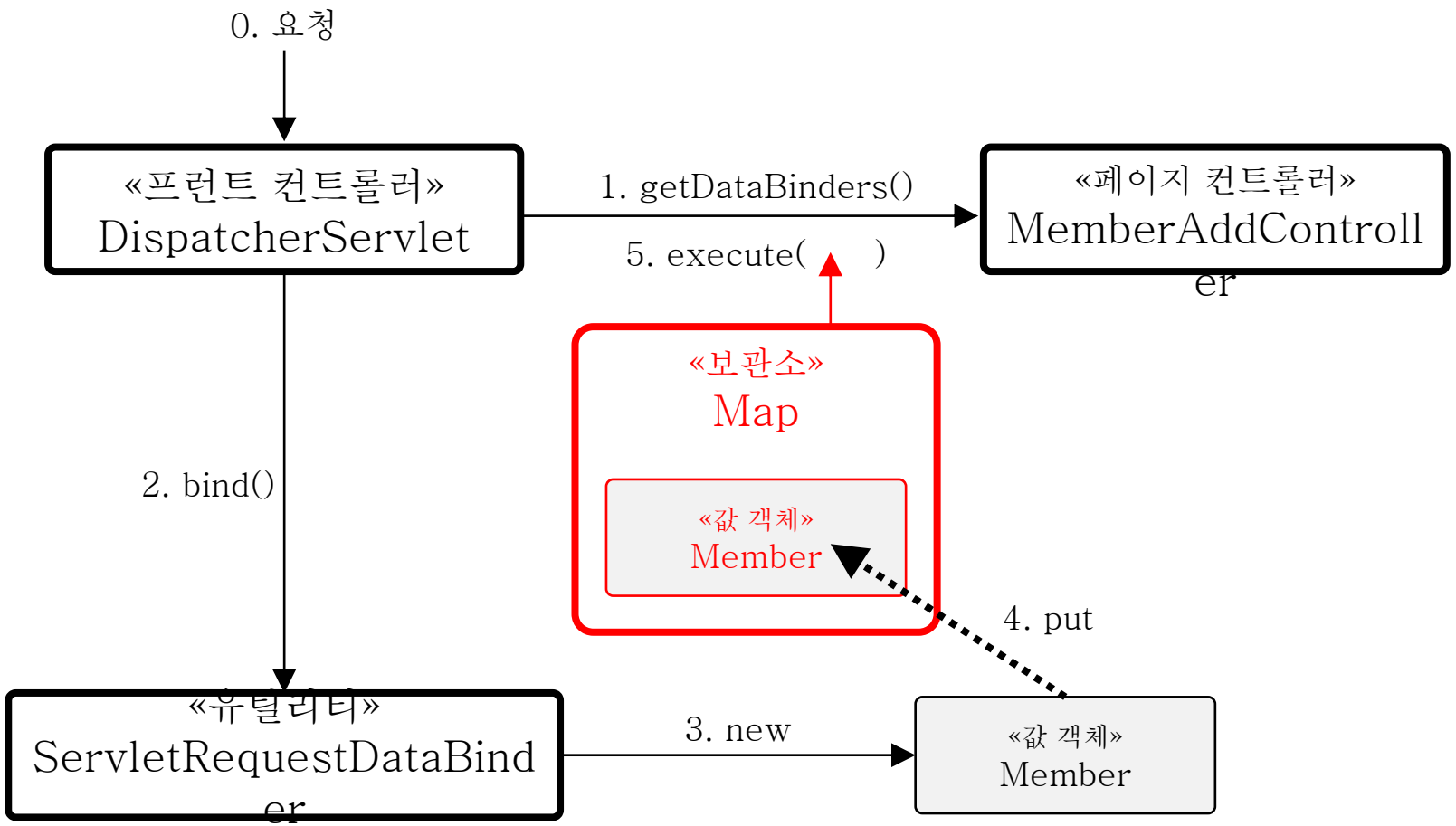
6.4 리플랙션 API를 이용하여 프런트 컨트롤러 개선하기

값 객체를 준비하는 과정



6.4 리플랙션 API를 이용하여 프런트 컨트롤러 개선하기

값 객체를 준비하는 과정



리플랙션 API 사용

6.4 리플랙션 API를 이용하여 프런트 컨트롤러 개선하기

인스턴스 생성하기

```
Member member = new Member();
```

리플랙션 API 호출

```
Class classInfo = spms.vo.Member.class;
Member member = (Member) classInfo .newInstance();
```

클래스에 선언된 메서드 정보 알아내기

```
Class classInfo = spms.vo.Member.class;
Method[] methodList = classInfo.getMethods();
Method method = classInfo.getMethod("getEmail");
```

6.4 리플랙션 API를 이용하여 프런트 컨트롤러 개선하기

메서드 이름 알아내기

```
Class classInfo = spms.vo.Member.class;
Method[] methodList = classInfo.getMethods();
Method method = classInfo.getMethod("getEmail");
```

```
for (Method m : methodList) {
  System.out.println(m.getName());
}
```

메서드 호출하기

```
Class classInfo = spms.vo.Member.class;
Member instance = (Member) classInfo .newInstance();
Method m = classInfo.getMethod("setEmail", String.class);
m.invoke(instance, "test@test.com");
System.out.println(instance.getEmail());
```

출력 결과

test@test.com