MVC 프레임워크 2단계 구현 실습

—

실습 환경

https://github.com/slipp/jwp-basic 저장소를 자신의 계정으로 fork한다.

- Fork한 저장소를 https://youtu.be/xid_GG8kL_w 동영상 참고해 로컬 개발 환경을 구축한다.
- jwp-basic 저장소의 **step5-qna-with-ajax** 브랜치로 변경한다.
  - 브랜치를 변경하는 방법은 https://youtu.be/VeTjDYI7UVs 동영상을 참고한다.
- src/test/java 디렉토리의 next.WebServerLauncher를 실행한 후 브라우저에서 http://localhost:8080으로 접속해 질문/답변 게시판 서비스 화면이 나타나는지 확인한다.

다음 두 코드의 문제점을 찾아보자.

```
public class AddAnswerController implements Controller {
    private static final Logger log = LoggerFactory.getLogger(AddAnswerController.class);

    @Override
    public String execute(HttpServletRequest req, HttpServletResponse resp) throws Exception {
        Answer answer = new Answer(req.getParameter("writer"),
                req.getParameter("contents"),
                Long.parseLong(req.getParameter("questionId")));
        log.debug("answer : {}", answer);

        AnswerDao answerDao = new AnswerDao();
        Answer savedAnswer = answerDao.insert(answer);
        ObjectMapper mapper = new ObjectMapper();
        resp.setContentType("application/json;charset=UTF-8");
        PrintWriter out = resp.getWriter();
        out.print(mapper.writeValueAsString(savedAnswer));
        return null;
    }
}
```

```
public class DeleteAnswerController implements Controller {
    @Override
    public String execute(HttpServletRequest req, HttpServletResponse resp) throws Exception {
        Long answerId = Long.parseLong(req.getParameter("answerId"));
        AnswerDao answerDao = new AnswerDao();

        answerDao.delete(answerId);

        ObjectMapper mapper = new ObjectMapper();
        resp.setContentType("application/json;charset=UTF-8");
        PrintWriter out = resp.getWriter();
        out.print(mapper.writeValueAsString(Result.ok()));
        return null;
    }
}
```

문제점

- JSON으로 응답을 보내는 경우 이동할 JSP 페이지가 없다보니 불필요하게 null을 반환해야 한다. AJAX에서 사용할 컨트롤러는 반환 값이 굳이 필요없다.
- AddAnswerController와 DeleteAnswerController를 보면 자바 객체를 JSON으로 변환하고 응답하는 부분에서 중복이 발생한다. 이 중복을 제거하고 싶다.

요구사항

이번 실습의 목표는 이 두 가지 문제점을 해결하도록 앞에서 구현한 MVC 프레임워크를 개선하는 것이다. 어떻게 구조를 개선하는 것이 앞으로도 확장 가능한 구조가 될 것인지 설계한 후에 구현을 시작해 보기 바란다.

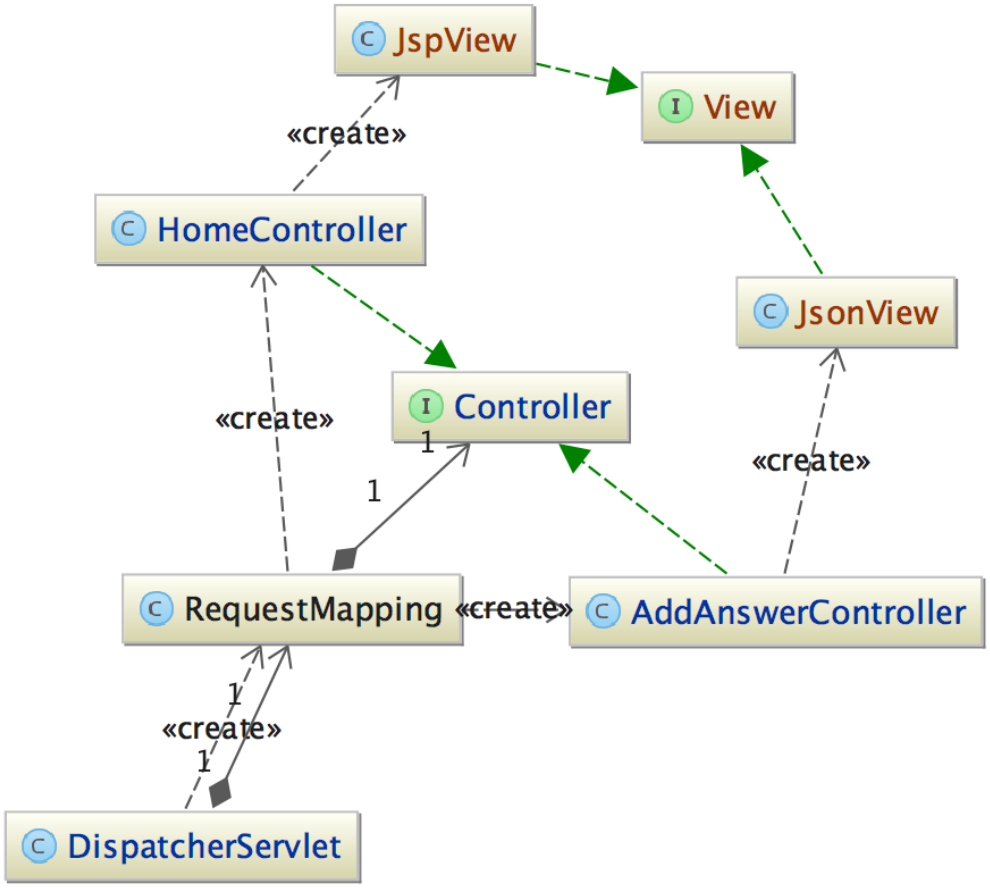
1단계 힌트

—

막상 개선하려니 막막한 독자들을 위해 내가 접근한 방식을 간단히 공유한다. 내가 접근한 방법은 View 인터페이스를 추가해 구조를 변경하는 것이다. 지금까지는 뷰가 JSP 하나였는데 JSON 응답을 담당하는 다른 종류의 뷰가 하나 추가되었기 때문에 이를 추상화한 View 인터페이스를 추가했다. **View 인터페이스를 구현하는 JspView, JsonView**를 추가해 컨트롤러에서 전달하는 모델 데이터를 활용해 응답하도록 개선했다. 현재 HttpServletRequest에 저장해 전달하는 모델 데이터를 HttpServletRequest를 사용하지 않고 별도의 Map을 활용해 전달하도록 구조를 변경했다. 이 모델 데이터와 뷰를 추상화한 ModelAndView를 추가해 Controller 인터페이스를 다음과 같이 변경했다.

```
public interface Controller {
    ModelAndView execute(HttpServletRequest request, HttpServletResponse response) throws
Exception;
}
```

2단계 MVC 프레임워크 구현 완료 후 클래스 다이어그램



요구사항 분리 및 힌트

—

실습

경우의 수가 증가하면서 if/else if/else 절이 계속 발생하는 상황은 앞에서도 경험했다. 앞에서 이 문제를 해결하기 위해 Controller 인터페이스를 추가했듯이 뷰를 추상화한 인터페이스를 추가한다.

힌트

- View라는 이름으로 인터페이스를 추가한다.

```
public interface View {
    void render(HttpServletRequest request, HttpServletResponse response) throws
Exception;
}
```

실습

View를 구현하는 JspView를 구현한다.

힌트

- JspView의 생성자는 이동할 URL을 인자로 받는다. 즉, Controller의 execute() 메소드의 반환 값을 가진다.
- JspView의 render() 메소드는 DispatcherServlet의 move() 메소드를 구현한다.

실습

View를 구현하는 JsonView를 구현한다.

힌트

- JsonView는 생성자로 인자를 전달하지 않아도 된다.
- JsonView는 자바 객체를 JSON으로 변환한 후 응답을 보내는 기능을 구현한다.
  - HttpServletRequest를 통해 전달하는 모든 값을 Map에 저장한 후 JSON으로 변환한다.

```
private Map<String, Object> createModel(HttpServletRequest req) {
    Enumeration<String> names = req.getAttributeNames();
    Map<String, Object> model = new HashMap<>();
    while (names.hasMoreElements()) {
        String name = names.nextElement();
        model.put(name, req.getAttribute(name));
    }
    return model;
}
```

  - 위와 같이 구현하는 경우 JSON 데이터 구조가 변경되기 때문에 자바스크립트에서 JSON 데이터 사용하는 부분을 수정해야 한다.

실습

Controller 인터페이스의 반환 값을 String에서 View로 변경한다.

실습

각 Controller에서 String 대신 JspView 또는 JsonView 중 하나를 사용하도록 변경한다.

실습

DispatcherServlet에서 String대신 View 인터페이스를 사용하도록 수정한다.

View를 인터페이스로 분리하는 것만으로 충분히 유연한 MVC 프레임워크가 될 수 있다. 이 단계에서 멈춰도 괜찮다. 여기서 멈추지 않고 한 단계 더 리팩토링을 한다면 하고 싶은 부분이 있는가?

HttpServletRequest 사용하면서 발생하는 하나의 이슈는 JsonView는HttpServletRequest에 추가되어 있는 모든 데이터를 JSON으로 변경한다. 그런데HttpServletRequest의 경우 서블릿 필터, 서블릿의 여러 단계를 거치면서 개발자가 모르는 상태에서 값이 추가되는 상황이 발생할 수 있다. 이 경우 개발자가 의도하지 않은 데이터가 불필요하게 JSON으로 변경되어 클라이언트 응답으로 보내질 수도 있다. HttpServletRequest를 통해 데이터를 전달하지 않고 개발자가 원하는 데이터만 뷰에 전달할 수 있도록 모델 데이터에 대한 추상화 작업을 진행해 본다.

실습

모델 데이터에 대한 추상화 작업을 진행한다.

힌트

- 모델 데이터를 View와 같이 전달해야 하기 때문에 ModelAndView와 같은 이름의 클래스를 새로 추가한다.
- ModelAndView는 View와 모델 데이터를 Map<String, Object> 형태로 관리하도록 구현한다.

실습

View의 render() 메소드에 모델 데이터를 인자로 추가하고 JspView와 JsonView 수정한다.

힌트

- View의 render() 메소드 인자에 Map을 추가한다.

```
public interface View {
    void render(Map<String, ?> model, HttpServletRequest request,
HttpServletResponse response) throws Exception;
}
```

- JspView의 render() 메소드는 모델 데이터를 꺼내HttpServletRequest에 전달한다.
- JsonView의 render() 메소드는HttpServletRequest 메소드에서 Map으로 변경하는 부분을 제거한다.

실습

앞 단계와 같이 Controller의 반환 값을 View에서 ModelAndView, 각 Controller 구현체는HttpServletRequest 대신 ModelAndView를 통해 데이터를 전달하도록 수정, DispacherServlet에서 View대신 ModelAndView를 사용하도록 수정한다.