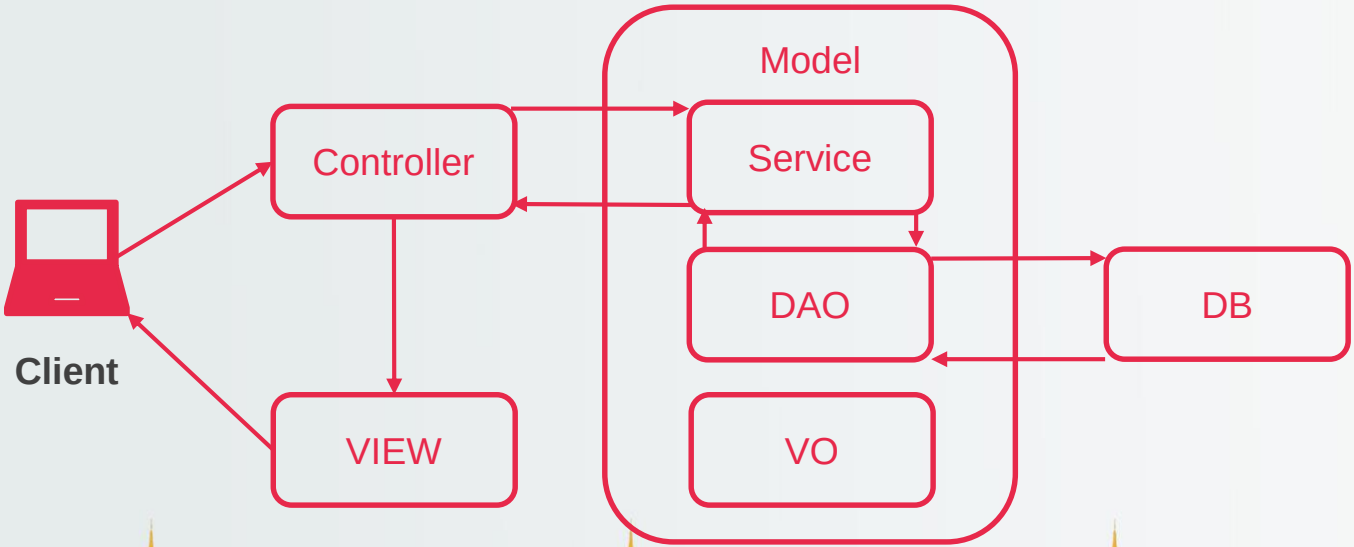
MVC 아키텍쳐

MVC

MVC

Model – View – Controller

- 웹 어플리케이션 개발 시 MVC 패턴을 적용하여 각각의 역할 별 작업이 가능하도록 분담하는 설계 패턴



MVC

Model

- 웹 어플리케이션의 비즈니스 로직, 수행할 서비스를 담당

1. Service : 여러 DAO를 호출하여 데이터 접근/갱신을 하고, 그렇게 읽은 데이터에 대한 비즈니스 로직을 수행하여 Controller에 그 결과를 전송하는 클래스
2. DAO : 데이터베이스에 직접 접근하여 요청 받은 결과를 반환하는 클래스
3. VO : 계층 간 데이터 교환을 위한 객체 클래스
   동의어로 DTO, Domain Object, Bean, Entity

MVC

View

- 사용자가 요청하거나 요청한 정보를 응답 받아 볼 수 있는 화면을 담당하며 JSP, HTML 등을 통해 표현

Controller

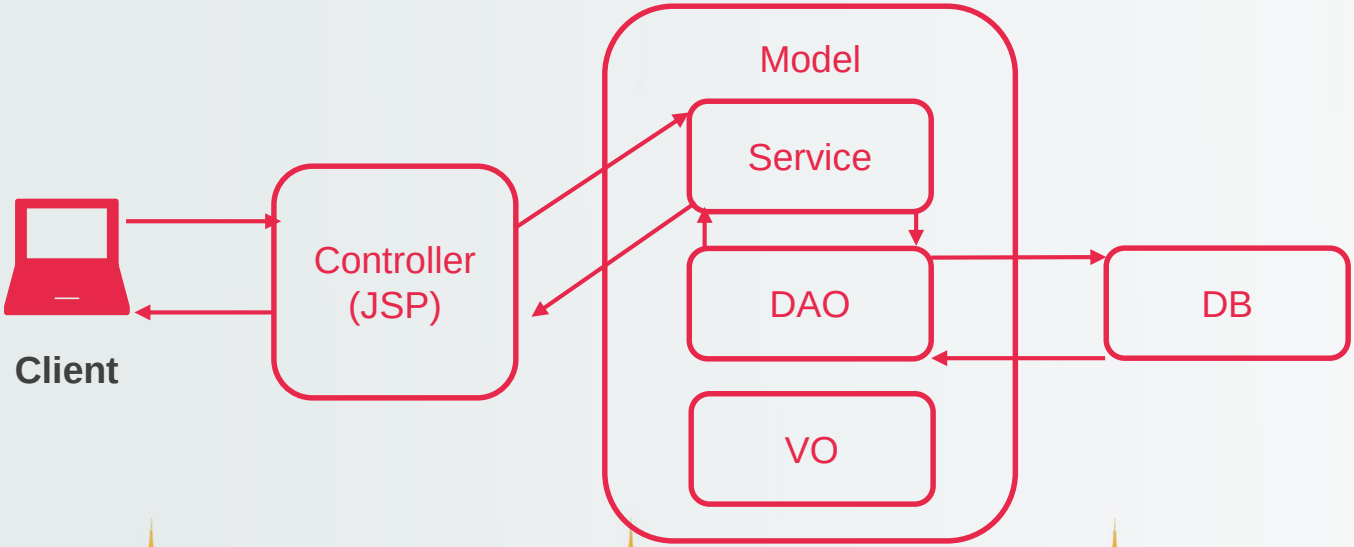
- 사용자의 요청을 전달 받아 응답 처리를 위한 Service를 호출하고 결과를 View에 전송하는 계층
- 전달 받은 정보를 바탕으로 사용자 요청을 분석한 후, 이를 서비스에 전달할 VO객체를 생성하여 전달하고 Service로부터 결과를 리턴 받아 관련된 View 화면에 응답
- MVC1 패턴에서는 JSP, MVC2 패턴에서는 Servlet이 담당

MVC 패턴

MVC 패턴

MVC1

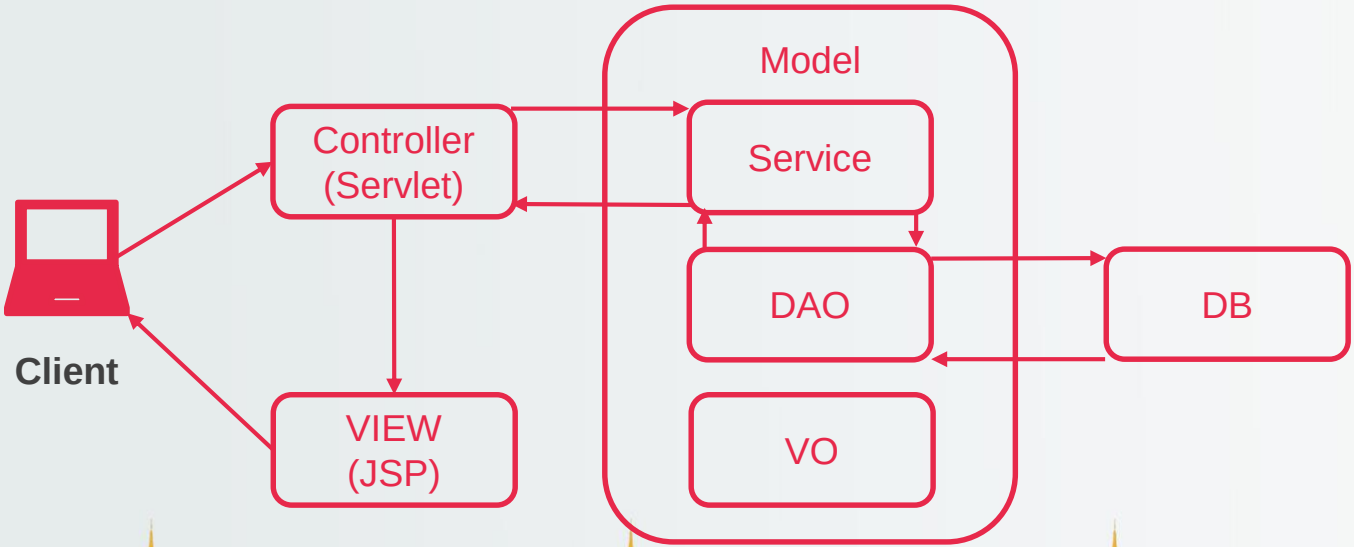
- jsp페이지가 사용자의 요청을 받아 비즈니스 로직을 수행한 후 해당 페이지에서 결과처리까지 진행하는 방식으로 소규모 프로젝트에 적합
  → Controller와 View가 같은 jsp페이지



MVC 패턴

MVC2

- 사용자의 요청을 받아 비지니스로직을 수행하는 역할은 Servlet이 하고 비지니스로직 수행결과를 통한 View는 JSP로 구현하는 방식



MVC 패턴

응답할 view 페이지가 동적 페이지인 경우

DispatcherServlet : 사용자의 요청을 다른 서블릿이나 JSP 페이지에 전달할 때 사용하는 클래스로 request 객체를 사용해 생성할 수 있다.

```
//request 객체를 통한 RequestDistpatcher 객체 생성(매개변수가 결과 페이지)
RequestDispatcher view = request.getRequestDispatcher("/result.jsp");
//결과페이지에 전달할 값 저장("member"라는 key값으로 membe객체 전달
request.setAttribute("member",member);
//request와 response를 인자값으로 전달하여 정보 전송
view.forward(request, response);
```

MVC 패턴

응답할 view 페이지가 정적 페이지인 경우

- 정적인 html의 경우에는 동적으로 화면을 생성할 필요가 없으므로 응답 view에 정보를 전달할 필요가 없다.
- 이때 사용하는 것이 HttpServletResponse 객체의 sendRedirect()

```
//클라이언트에게 /error.html 페이지를 요청하도록 함
response.sendRedirect("/error.html");
```