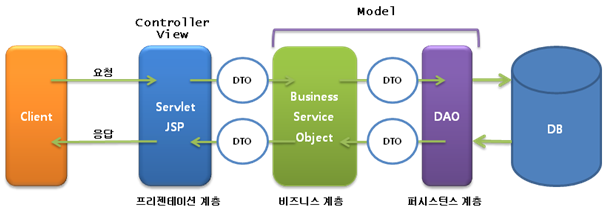
**MVC 디자인 패턴**

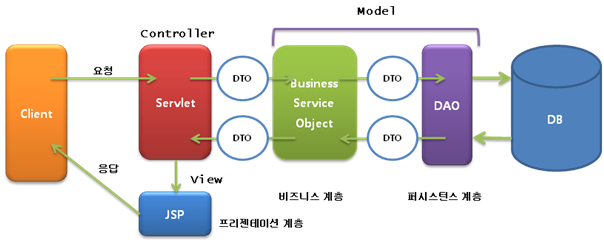
|  |
| --- |
| MVC 패턴이란 **애플리케이션을 크게 모델(Model, View, Controller)의 세 영역으로 구분하고 영역 간의 결합도를 최소화** 패턴이다. 주로 Web Application에서 사용되며, 일반 애플리케이션에서도 널리 사용된다.  MVC 패턴의 가장 중요한 장점 중 하나는 비즈니스 로직과 프리젠테이션 로직의 분리이다. 이로서 디자이너와 개발자들이 각자의 영역에 집중할수 있게 되었다.(효율성 극대화) |

■ Model 영역   
**애플리케이션에 비즈니스 로직과 사용되는 데이터를 다루는 영역이다.**일반적으로 비즈니스 데이터는 DB에 의해 관리되고, 그 데이터를 다루는 연산은 SQL문을 통해서 구현된다.  
  
■ View 영역   
**최종 사용자에게 보여줄 프리젠테이션 로직을 담당하는 영역이다.(**화면)  
일반적으로 자바 웹 애플리케이션에서는 JSP를 통해서 구현된다.  
  
■ Controller 영역  
컨트롤러는 흐름을 관리하는 역할을 하며, 모델과 뷰 영역간의 조정 역할을 한다.   
사용자의 요청을 받아 이를 수행하기 위한 비즈니스 로직을 선택하고 호출하며, 수행한 비즈니스 로직의 결과를 보여주기 위해 뷰를 선택하여 포워드 해주는 역할을 한다.   
  
  
**MVC 모델1 아키텍처**  
\* DTO(Data Transfer Object ) : 데이터 전송을 위한 개체



요청 흐름을 제어하기 위한 Controller 가 특별히 존재하지 않는다. MVC로 굳이 구분하자면 클라이언트의 요청을 받은 JSP가 Model 영역을 선택해서 호출하므로 Controller 역할을 수행한다고 볼 수 있다. JSP는 JavaBean을 이용하여 DB를 연동하여 Model 영역의 수행 결과를 받아 클라이언트에게 보내줄 UI(User Interface)를 생성하므로 View 역할도 수행한다.   
프리젠테이션로직과 비즈니스로직의 분리가 어렵고 그로인해 뷰의 재활용성이 떨어진다. 하지만 개발기간이 짧은 소규모 Project나 변경사항이나 요구사항이 적을 경우 빠르게 개발할수 있다는 이점이 있다.

**MVC 모델 2 아키텍처**



모델2에서는 서블릿이 흐름을 제어하는 Controller 역할을 수행한다. 모든 클라이언트에 요청은 서블릿에 전달되어 서블릿이 클라이언트의 요청에 따라 비즈니스 로직을 호출하고 그결과에 따라 View에 해당하는 JSP를 호출하는 방식이다.

모델2가 모델1에 비해 개발 초기의 부하가 비교적 높고, 아키텍처에 대한 수준 높은 이해를 요구하지만, **재사용성이 높고 확장이 모델1에 비해 용이하다는 장점이 있다.** 무엇보다, 모델2는 그 구조적 특징으로 인해 자연스럽게 MVC각 영역의 역할이 구분되도록 해준다.

모델2는 Controller 역할을 하는 서블릿의 역할이 매우 중요하며, **서블릿이 잘못 개발된 경우 애플리케이션 전체에 미치는 영향이 매우 크다.** 초기에 모델2를 적용한 웹 애플리케이션 중에는 서블릿 구현시 Model 영역과 View 영역을 호출해주는 부분이 하드 코딩되어 있거나 비효율적으로 구현되어 애플리케이션의 효율을 떨어뜨리고 구조만 복잡하게 되어버리는 경우가 많았다. 그러나 수많은 시행착오 끝에 잘 구축된 Controller들이 재활용되기 시작하면서 웹 **프레임워크(Framework) 형태로 보급**되기 시작하였다.

Model2 : MVC 패턴을 적용하여 **View는 JSP**로 **Controller는 Servlet**으로 **Model은 bean(=class**)으로 작성