웹 애플리케이션 아키텍쳐

웹 애플리케이션 아키텍쳐의 이해를 위해 MVC 패턴과 애플리케이션 레이어 그리고 각 각의 레이어에 대한 역할을 알아보자

1 MVC 패턴

MVC 패턴은 Model-View-Controller 패턴을 줄인 것으로 제록스 연구소의 트뤼그베 린즈커그(Trygve Reenskaug)가 처음으로 소개한 개념으로 관심사의 분리(SoC, Separation of Concerns)를 통한 업무의 분할을 위한 디자인 패턴 이다.

MVC 패턴은 원래 데스크톱 애플리케이션에서 사용하기 위해 디자인 되었다. 프레젠테이션 레이어를 여러 컴포넌트로 분리해 각 컴포넌트가 특정 기능을 담당하게 한다. 먼저 렌더링된 뷰를 통해 사용자는 컨트롤러로 액션을 요청한다. 컨트롤러는 모델을 업데이트 하고, 모델은 뷰에게 렌더링을 요청한다. 이는 다시 사용자에게 응답된다.

MVC 컴포넌트의 기능

컴포넌트	기능
컨트롤러	폼 액션이나 링크 클릭등을 통하여 사용자가 요청하는 액션을 받아들여서 모델과 뷰를 통해서 요청을 처리한 후 응답한다.
모델	모델은 뷰가 렌더링하는데 필요한 데이터이다. 예를 들면 게시판 리스트에 출력하기 위한 게시물 목록이 모델에 해당한다.
뷰	실제로 유저에게 보여주기위한 화면이며 모델을 이용하여 렌더링을 한다. JSP, XML, JSON, PDF 등으로 웹 페이지를 표현한다.

웹 환경에서는 HTTP 프로토콜의 특성으로 모델에서 변경된 내용을 바로 뷰로 전달하지 않고 서버로부터 변경 내역을 가져와야 한다. 웹은 설계 구조상 무상태이기 때문에 요청 작업을 처리하는 동안 모델을 유지하는게 쉽지 않다. 웹에서 MVC 패턴은 모델 2 아키텍쳐로 구현된다.

MVC 패턴과 모델 2 와의 차이는, 모델 2 는 프론트 컨트롤러를 두어 서버로 들어오는 요청을 다른 컨트롤러로 전달한다는 점이다. 요청을 전달받은 컨트롤러는 해당 요청을 처리하기 위해 모델을 만들고 뷰를 선택한다.

스프링 MVC 에서는 DispatcherServlet 이 프론트 컨트롤러의 역할을 담당하며 이를 통해 비즈니스 로직 및 모델 데이터를 담당하는 개발자는 사용자 UI 에 대한 부담을 덜 수 있고, 뷰를 담당하는 웹디자이너는 비즈니스 로직과 모델 데이터에 대한 부담을 덜 수 있다. 이렇게 업무를 분리하므로서 애플리케이션의 유지 보수와 테스트가 수월해진다.

2 티어와 레이어

티어는 배포시점을 기준으로 한 물리적인 개념이며 레이어는 애플리케이션을 관심사별로 분리한 논리적인 개념이다.

티어는 클라이언트 층, 중간 층, EIS(Enterprise Information System) 층으로 된 3 개 층이 기본이다.

애플리케이션별 티어

애플리케이션	클라이언트 층	중간 층	EIS 층
웹 애플리케이션	웹 브라우저	애플리케이션 서버	DB, 레거시 시스템
스마트폰 앱	스마트 폰		

레이어는 주로 중간 층에 있는 웹 애플리케이션을 논리적으로 분리한 것이다.

레이어	역할
프레젠테이션	사용자 인터페이스(UI)와 컨트롤러를 제공한다. 주로 Controller 혹은 Action 이 붙은 클래스들로 화면 전환이나 화면의 버튼 이벤트에 대한 액션, 세션 관리 등을 한다.

서비스	비즈니스 로직을 제공한다. 주로 Service 가 붙은 클래스들로 특정 업무에 대한 처리, 최소 업무 단위(트랜잭션)에 대한 처리 등을 한다.
데이터 액세스	데이터베이스 액세스를 추상화 한다. 주로 Dao 가 붙은 클래스들로 시스템이 사용하는 데이터 액세스 기술을 상위 레이어에 노출하지 않고 접근하게 해준다.

웹 MVC 애플리케이션 레이어는 위 레이어에 User Interface 와 Domain Model 이 추가된 형태이다.

Domain Model : DTO, VO, Bean ...	User Interface : View (jsp, velocity, x-platform...)
	Presentation : @Controller
	Service : @Service
	Data Access : @Repository

도메인 레이어(VO, DTO, Bean ...)

서비스로부터 비즈니스를 실행하는데 있어 고객이나 주문같은 클래스의 집합을 도메인 오브젝트라고도 한다.

유저 인터레이스 레이어(View)

애플리케이션을 사용자에게 보여주는 역할을 한다. 다양한 뷰 기술이 있는 상태에서 코드를 최대한 재 활용하기 위하여 프레젠테이션 레이어를 유저 인터페이스 레이어와 웹 레이어로 나뉜다.

웹 레이어(Presentation Layer)

서버로 들어오는 요청을 서비스 레이어에서 다룰 수 있게 변환하고, 서버에서 생성한 결과를 유저 인터페이스에서 사용가능한 응답 결과로 바꿔주는 역할을 한다. 여기에 스프링 MVC 를 사용할 수 있다.

서비스 레이어

사용자에게 시스템의 기능을 노출한다. 사용자가 다양한 클라이언트를 사용하더라도 비즈니스 로직은 동일해야 한다. 따라서 단일 진입점과 대단위 메소드를 두어야 하며 이를 이용하여 트랜잭션과 보안을 쉽게 적용할 수 있다.

웹 기반 환경에서는 많은 사용자가 동시에 서비스를 요청하는 경우가 있다. 서비스는 상태를 갖지 않아야 하기 때문에 서비스는 싱글톤으로 만드는 것이 좋다. 상태 유지는 도메인 모델에 맞긴다.

데이터 액세스 레이어

데이터 액세스 레이어는 시스템의 퍼시스턴스와 데이터를 주고받는 역할을 한다. 서비스 레이어는 데이터 저장소가 무엇인지 모르지만 데이터 액세스 레이어는 데이터 저장소에 데이터를 저장하고 검색하는 방법을 알고 있다.

위에 소개한 웹 MVC 애플리케이션 레이어 이외에도 다양한 분류 방법이 있으며 프로젝트나 이용 기술에 따라 구분해 사용해야 한다.

그 외 다양한 레이어 구조들

<table>
<tr><th>레이어 1</th><th>레이어 2</th><th>레이어 3</th><th>레이어 4...</th></tr>
<tr><td rowspan="2">프레젠테이션</td><td rowspan="2">프레젠테이션</td><td>프레젠테이션</td><td>클라이언트 층에 있는 프레젠테이션</td></tr>
<tr><td>애플리케이션 컨트롤러</td><td>중간 층의 프레젠테이션</td></tr>
<tr><td rowspan="2">비즈니스</td><td>서비스</td><td rowspan="2">도메인</td><td rowspan="2">비즈니스</td></tr>
<tr><td>도메인</td></tr>
<tr><td>데이터 액세스</td><td>퍼시스턴스</td><td>인티그레이션</td><td>인티그레이션</td></tr>
<tr><td colspan="3"></td><td>리소스</td></tr>
</table>