프레임 워크

1. 프레임 워크의 개념

소프트웨어의 구체적인 부분에 해당하는 설계와 구현의 재사용성을 위해서 일련의 협업화 된 형태로 클래스 들을 제공하는 것. 시스템 개발에 일관적인 틀을 주어서 개발된 SW에 통일성을 보장해 줌. 덕분에 개발 과정이 간단해지지만 자유가 제한됨.

핵심 로직에만 집중할 수 있도록 기본적인 기능(라이브러리)을 제공해 줌과 동시에 틀을 갖추고 있어 프레임워크가 정의한 틀에 맞춰 기능을 구현해야 함.

쉽게 말하자면 뼈대. 라이브러리\*와 차이점은 프레임워크는 라이브러리에 기본 틀(설계된 구조와 그에 따른 규칙)에 더해 작업을 호출할 수 있는 제어권이 추가된 개념이라고 생각하면 된다.

1. 장점

* 효율적

시간, 비용 절약, 생산성 증대.

* 퀄리티 향상

버그 발생 가능성을 처리해줌+다수 사용자에 의한 검증. 일관된 품질 보장.

* 유지보수가 용이하다

담당자가 바뀌어도 코드가 체계적이기 때문에 유지보수가 안정적이다.

1. 단점

* 학습시간이 길다

코드를 습득하고 이해하는데 시간이 걸린다

* 제약 사항

자유롭고 유연한 개발에 한계가 있다.

* 라이브러리

자주 사용되는 로직을 재사용하기 편하게 모듈화 한 클래스의 집합.

쉽게 말하면 부품. 자주 쓰는 기능들이 구현되어 있어 원하는 개발에 끌어 쓰면 된다. 구조 설계나 틀이 따로 정해져 있지 않아 사용자가 직접 본인의 구조 설계를 해야 되지만 그만큼 자유도는 높다.

1. 프레임 워크의 종류와 특징

프레임 워크는 언어와 구조에 따라 다양한 종류가 존재한다. 각각의 프레임 워크의 특징과 구조 장, 단점을 이해하면 특정 프로젝트를 진행할 때 어떠한 프레임 워크를 사용하면 될지 결정하고 프레임워크를 이용하여 개발하는데 도움이 된다.

* 프레임 워크 선택 기준

프레임 워크를 선택하는 기준 제시된 기준 이외에 다양한 기준들이 존재할 수 있다.

• 학습 비용(시간, 노력 등)

• 생산성(프레임워크의 목적/기원, 의견제시 여부(잘못된 부분 알림, 적합한 구조 추천 등), 라이브러리 포함 여부, 프레임워크의 개발 및 설계 사상)

• 프레임워크 및 프레임 워크에 기반이 되는 언어의 성능(기반이 되는 언어의 장, 단점을 대부분 따라가기 때문에 언어 또한 중요하다)

• Caching 지원

• 확장성

• 보안

1. 웹 프레임워크

* JAVA 프레임워크

1. 스프링 프레임워크

JAVA 오픈소스 어플리케이션 프레임워크. Enterprise급 애플리케이션(기업을 대상으로 하는 애플리케이션)을 개발하기 위한 기능들을 모두 제공하지만 경량화 된 솔루션으로 Enterprise급 애플리케이션을 대상으로 하는 만큼 매우 큰 데이터 처리와 다수의 Transaction처리가 가능하다. IoC\*기반 프레임워크로써 경량 컨테이너를 사용해서 자바 객체의 생성, 소멸 등 라이프 사이클 전반을 관리하며 유저는 언제든지 관리되고 있는 객체를 요청하여 사용할 수 있다.

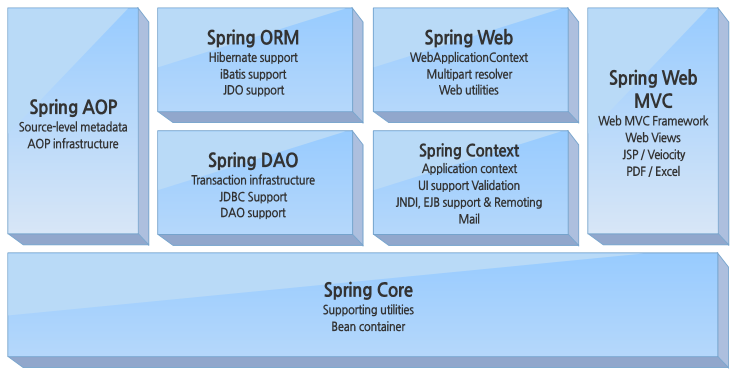
Spring은 IoC컨테이너를 DL\*, DI\*개념을 이용하여 구현했으며 이와 관련된 기능들은 제공한다.

EJB보다는 POJO를 권장 및 지원한다. POJO는 EJB에 비하여 의존성이 낮고 테스트가 단순하며 유지보수가 편해진다는 장점이 있다.

소프트웨어 개발 프로세스에 AOP(Aspect Oriented Programming, 관점 지향 프로그래밍)방식을 사용한다. AOP는 OOP와 대응되는 방식으로 OOP는 관심사 중심으로 데이터와 로직을 분리하고 캡슐화 하는 방식이었던 데에 반해 AOP는 OOP방식에 더해 핵심 기능(비즈니스 로직)과 공통 기능을 나눠서 관리하며 핵심 기능의 로직에 영향이 없게 공통기능을 끼워 넣는 방식으로 구현된다. 이러한 방식은 중복되는 코드를 제거하여 개발 효율을 높일 뿐 아니라 공통기능이 한곳에서 관리되기 때문에 수정과 유지보수가 용이하다.

MVC(Model2)디자인 패턴을 사용한다. MVC 패턴이란 UI와 비즈니스 로직을 분리하여 개발하는 패턴으로 각각의 부분이 나뉘어 있기 때문에 특정부분(예를 들어 View 부분)을 수정하는데 다른 부분에 영향이 없다. 그렇기 때문에 유지보수가 간단해지고 또한 각각의 부분을 재사용하기 용이하다는 장점을 가진다.

스프링의 구조는 아래와 같다



스프링 Core: 스프링 프레임워크의 핵심. 빈 팩토리 컨테이너가 안에 존재하며 객체 구성부터 의존성 처리까지 전반의 역할을 맡아 수행한다.

스프링 Context: Context 정보들을 제공하는 설정파일. 엔터프라이즈 서비스들을 포함하고 있다.

스프링 AOP: AOP프로그래밍을 할 수 있도록 도와주는 모듈.

스프링 DAO(Date Access Object): 데이터에 접근하는 객체. 추상 레이어를 지원하여 코딩 및 예외처리를 단순화할 수 있게 지원.

스프링 ORM(Object Relational Mapping): 객체와 테이블 간의 관계(매핑)를 설정하는 것. Ibatis 등의 OR도구를 사용할 수 있도록 지원.

스프링 Web: Web기반의 응용프로그램에 대한 Context를 제공하는 방식으로 Web Application개발에 필요한 기능을 지원.

스프링 MVC(Model-View-Control): MVC구조로 Application을 만들 수 있도록 지원. MVC 프레임워크는 전략 인터페이스를 통해 고급 구성이 가능하다.

• IoC

제어의 역전(Inversion of Control)의 약자. 기존 프로그램 들은 사용자가 객체의 생성, 호출 등 라이프 사이클과 제어의 전반을 관리하고 있었으나 이러한 제어권이 다른 객체나 컨테이너에게 옮겨진 프로그래밍 모델을 말함.

스프링의 경우에는 설정 메타 데이터에 필요한 객체와 의존성에 관해서 명시하여 클래스에 어떤 의존성이 필요한지 알 수 있게 한다. 기존에는 applicationContext.xml에 이러한 메타 데이터를 저장하는 방식이 많이 쓰였으나 점점 xml보다는 Annotation기반이나 자바 기반으로 설정하는 추세이다.

* DI

의존성 주입(Dependency Injection)의 약자. 객체를 외부에서 생성하여 주입시켜 주는 방식.

이렇게 되면 객체 간의 의존관계를 외부의 객체나 컨테이너 등이 관리하게 된다. 그 결과로 불필요한 의존 관계가 줄어들게 되어 결합도가 줄어들어 관리 수정 및 관리가 용이하다.

스프링에서는 클래스 사이의 의존관계를 빈(Bean = IoC컨테이너에 의해 관리되는 객체) 설정정보를 바탕으로 컨테이너가 자동으로 연결해 주는 식으로 구현되어 있다.

DI는 아래의 3가지 조건이 충족되어야 한다

* 클래스 모델이나 코드에는 동작시의 의존 관계가 드러나지 않아야 한다.
* 동작시의 의존 관계는 컨테이너 등 제3자가 결정해야 한다.
* 의존 관계는 사용할 객체의 레퍼런스를 외부에서 제공해 줌으로 써 만들어 진다(내부에서 객체를 생성하거나 직접적인 명세가 있어서는 안됨).
* DL

의존성 검색(Dependency Lookup)의 약자. 컨테이너가 관리하는 객체들을 컨테이너가 제공하는 API들을 이용하여 검색하는 방식을 말함.

스프링에서는 컨테이너 안에 있는 Bean에 접근하기 위해서 컨테이너가 제공하는 API를 이용하는 식으로 구현되어 있다.

* 파이썬 프레임워크

1. 장고 프레임워크

신문사 웹사이트 개발을 위한 프레임워크에서 시작하여 발전된 프레임워크로 현재는 보안이 우수하고 유지보수가 편리한 파이썬 기반의 프레임워크이다. 무료 오픈소스로 제공되며 커뮤니티들이 활발하게 유지되고 있다. 반면 Python을 사용하기 때문에 비교적 학습이 쉽고 개발속도나 코드 완성도가 높아지지만 Python의 단점(인터프리터 언어라서 실행시까지 에러 검출이 힘듦, 실행시간이 길다)을 계승한 점은 아쉬운 점이다.

다른 일반적인 프레임워크들과 달리 MTV라는 디자인 패턴을 사용한다. MTV는 Model(Model과 유사) – Template(View와 유사) – View(Controller와 유사)의 약자이다. MTV 패턴은 클라이언트의 URL요청을 토대로 Model에서 데이터에 접근하여 View에서 처리한 뒤에 Template에서 HTML파일을 읽어와 Render한 뒤 클라이언트에게 돌려주는 방식이다.

Batteries included라는 철학을 가지고 구현돼 있기 때문에 대부분의 기능들이 Modulization된 라이브러리로 지원된다(로그인, 인증 등). 덕분에 개발 시간을 절약하고 생산성을 증가시킬 수 있다. 반대로 개발자의 자유도가 떨어지는 단점이 있다.

장고는 보안에 관련된 라이브러리도 기본적으로 제공되기 때문에 보안에 강점이 있다. 예를 들어 유저의 계정, 비밀번호 관리에 관한 부분에 더하여 SQL인젝션, 요청 위조 등의 공격에 대비하는 부분까지 기본적으로 제공된다.

Shared-nothing아키텍쳐를 사용하기 때문에 각 부분은 독립적이다. 그렇기 때문에 각 부분을 교체, 변경하기 쉽고 이는 쉬운 수정 및 유지보수로 이어진다.

ORM을 지원한다. 그렇기 때문에 다양한 데이터 베이스 시스템을 선택 및 변경하기 용이하다.

자체 템플릿 시스템을 가지고 있어서 화면 디자인 및 로직에 대해서 독립적으로 작업할 수 있다. 또한 이 시스템은 HTML과 같은 텍스트형 언어를 쉽게 다룰 수 있게 지원한다.

캐시 시스템이 존재한다. 자주 이용되는 페이지나 내용을 캐시에 저장해 뒀다가 재사용하는 방식은 서버 성능과 유저가 체감하는 반응속도에 큰 영향을 준다. 이런 캐시 시스템은 메모리, DB, 파일시스템 등 개발자가 원하는 곳에 위치시킬 수 있다. 또한 캐시 크기를 페이지부터 전체 서버까지 개발자가 원하는 단위로 지정할 수 있다.

장고는 코드의 변동사항이 자동 반영된다. Python 파일의 변경여부를 감시하고 있다가 변경시에 바로 적용해 준다. 그렇기 때문에 업데이트 시에 서버를 내리고 다시 올리는 동작이 필요가 없다.

장고 프레임워크의 동작과정은 아래와 같다.



유저의 요청이 들어오면 urls.py에서 어떤 요청인지 분석하여 알맞은 View로 연결해 준다. View에서 요청에서 필요한 처리를 하는데 이 때 필요한 데이터는 Model에 서 얻어와 처리한다. 이후 Template에서 동적으로 HTML페이지를 만들고 View에서 넘겨준 데이터를 뿌려 줌으로써 유저에게 돌려줄 페이지를 완성한다. 이후 Template는 다시 View에게 이 HTML페이지를 돌려주고 View는 유저에게 이 페이지를 반환한다.

* PHP 프레임워크

1. Laravel 프레임워크

2011년에 출시된 PHP기반 프레임 워크이다. 현재 PHP프레임워크 중에서 가장 널리 쓰이는 프레임워크이자 오픈소스 프레임워크이다. 가장 특징적인 점은 기존 PHP에서는 불편했던 단위 테스트를 개선하기 위해 특별한 테스트 기능을 지원한다는 것이다. 이 테스트 기능 덕분에 테스트가 편하고 강력하다. 또 Artisan에서 제공하는 다양한 커맨드 명령어(Migration\* 등)가 존재하는데 이는 생산성 향상에 도움을 준다. 또한 한글화된 잘 정리된 문서가 많은 점도 장점이다.

MVC 디자인 패턴을 사용한다. MVC는 여타 MVC 패턴과 흡사하다.

데이터베이스 추상화 레이어가 존재한다. PDO(PHP Data Object)라고 부르는 데 이런 추상 레이어가 존재하기 때문에 특정 데이터베이스에 의존적이지 않다는 장점이 있다. 또한 DI를 사용하여 런타임때 사용할 DB를 결정하기 때문에 간단한 설정 변경만으로 DBMS의 종류를 변경할 수 있다.

프레임워크 자체적으로 보안에 관한 여러가지 기능을 제공하기 때문에 취약점들에 대한 대비가 상대적으로 쉽다. 또한 보안을 고려한 설계가 되어있는 점 또한 강점이다. 예를 들어 Laravel의 경우 설정파일이 숨겨져 있어 외부에서 접근할 수 없고 public폴더 또한 최소한으로 노출되게 설정되어 있다. 이외에도 SQL인젝션, 크로스 사이트 Scripting 등에 대한 대비도 되어있다.

템플릿 엔진이 내장 되어있다. 덕분에 뷰 코드 작성시에 MVC패러다임에 맞는 코딩이 가능하다. 내장되어 있는 템플릿은 블레이드 템플릿으로 상속을 지원해 공통적인 부분을 Modulization하여 사용하거나 관리하는 것이 가능하다. 또한 HTML과 CSS구문 사이에 PHP구문만 특정 태그로 삽입할 수 있다.

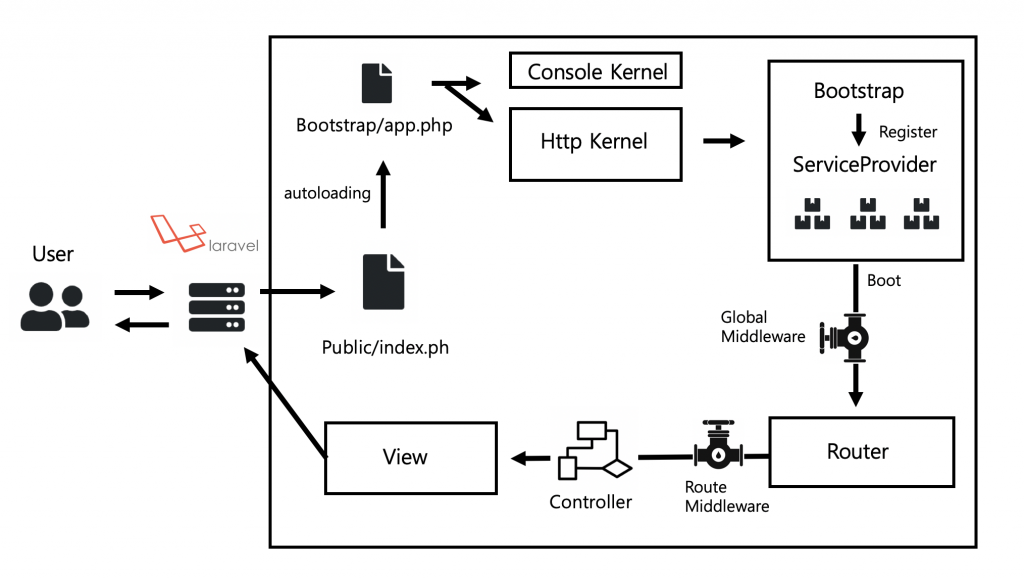
Pretty URL을 이용해 파일시스템과 분리된 URL로 서비스를 제공할 수 있다. 일반적인 URL은 파일시스템 + GET 파라미터 값으로 보여 직관적이지 않고 지저분하다. 라라벨에서는 이런 문제가 개선된다.

DI와 IoC개발 패턴이 적용 되어있다. 라라벨은 생성자를 이용하는 방법으로 이를 구현하였다. 런타임에 DI가 이루어지며 Provider는 설정파일에 지정되어 있다. 이러한 구조는 설정파일의 변경만으로 Application변경없이 구조를 바꿀 수 있는 장점을 가진다.

CoC(Convention over Configuration)는 설정보다 관례를 우선시한다는 뜻으로 수많은 설정파일의 내용을 정해진 관례에 따라 처리하도록 하여 개발자가 일일이 설정해 주어야 하는 부담을 줄여준다. 하지만 개발자의 자유도가 제한된다는 단점이 존재한다.

PHP의 경우 단위테스트나 기능테스트가 어려웠던 점이 있다. 이런 점을 해결하기 위해 PHPUnit을 지원한다. 이를 이용하면 코드가 수정되었을 때 마다 자동으로 단위테스트를 진행하고 이상이 없을 경우 다른 환경에 배포하는 등 업무 자동화에 도움을 준다.

Laravel 프레임워크의 동작과정은 다음과 같다.



요청이 들어오면 index.php가 동작한다. Index.php에서는 Boostrap/app.php 스크립트를 참고해 라라벨 어플리케이션의 인스턴스를 가져오며 Service Container를 생성한다. 이후 요청은 HTTP 혹은 Console커널로 이동되어 전 처리 과정(에러처리, 로그설정 등)을 거친다. 이후 Application 커널이 Bootstrap되는데 이 과정에서 서비스 Container에 서비스 Provider를 로드하고 등록하는 과정도 포함된다(config/app.php에 등록된 모든 서비스 Provider가 로드되는 것처럼 보이나 내부적으로는 요청이 들어올 때 필요한 서비스 Provider만 로드되고 나머지는 필요한 때에 로드 된다). 서비스 Provider는 서비스 컨테이너에 바인딩을 등록하는 것을 포함해서 이벤트 Listener, 미들웨어 그리고 Router등을 등록한다. 이를 Bootstraping(쉽게 말해 깨워서 데리고 옴)이라고 한다. 이후 요청은 라우터로 전달되어 알맞은 컨트롤러로 Dispatch되어 처리된다. 이후 생성된 View가 사용자에게 전달되는 방식으로 구현되어 있다.

• 마이그레이션

Laravel에서 지원하는 기능으로 Artisan명령어를 이용하여 생성한 마이그레이션 파일안에서 스키마 Builder를 사용하는 방법으로 간접적으로 데이터베이스 및 데이터 베이스 스키마를 변경할 수 있게 해준다(테이블, 컬럼, 인덱스추가 등).

• Composer

Composer는 현대적인 PHP 프로그래밍에서 빼 놓을 수 없는 의존성 관리자다.

Composer를 사용하면 PSR-0, PSR-4를 지원하는 외부 라이브러리와 프레임워크를 편리하고 자유롭게 가져다 사용할 수 있다. 또한 외부 라이브러리를 많이 사용하면 겪게 되는 문제인 Dependency Hell 문제를 해결할 수 있다.

+ 의존성 관리자란? 일반적으로 패키지 관리자가 그 역할을 하는 경우가 많다. 패키지 관리자의 역할에 더해서 패키지가 요구하는 다른 패키지들을 거슬러 올라가면서 설치해주며 동시에 Dependency Tree를 만들고 관리해 주는 역할을 한다.

+ 사용 예) 외부에서 개발한 Laravel패키지를 내 Application에 추가하여 사용할 경우 Composer를 통해 패키지를 설치하고 config/app.php의 Provider항목에 등록해 주면 된다.

* 자바 스크립트 프레임워크

1. Vue.js

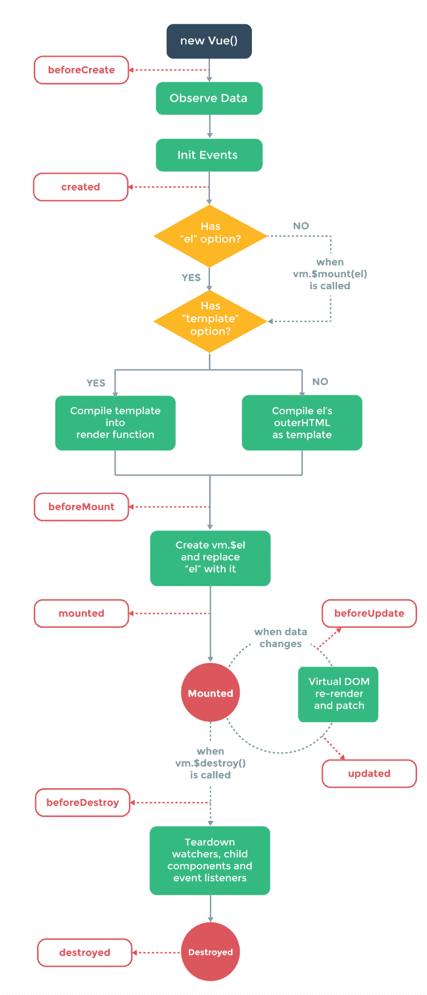
2014년에 릴리즈 된 자바스크립트 오픈소스 프레임워크이다. View에 최적화된 프론트엔드 프레임워크로 기존 MVC패턴이 아닌 MVVM(Model-View-ViewModel)패턴\*으로 디자인되었다. MVVM패턴이 사용된 이유는 데이터와 프레젠테이션 로직을 분리하기 위함으로 이를 통해 테스트 및 유지보수가 용이해지고, 재사용성이 좋아진다. 또한 컴포넌트를 사용하여 UI들을 묶고 뷰 레이어를 정리할 수 있다.

Vue.js는 템플릿 코드만 수정하여 개발하기를 권장한다. 개발자가 Template코드를 수정하면 Vue에 내장된 render()함수가 이 템플릿을 JS코드로 변환해주고 또한 이것을 가상 DOM\*에 추가해 주는 방식으로 이를 가능하게 하였다.

Vue.js 는 크게 3가지로 나뉘어져 있다.

1. Vuex: Vuex는 가상 DOM을 관리하는 매니저 역할을 한다. 자세히 설명하면 앱 또는 웹의 요소들의 변경을 모니터링하고 변경시에 그 값들을 업데이트 해주는 일을 하는 것이다.
2. Vue CLI: Vue CLI는 Vue Command Line Interface의 약자이다. 프로젝트를 생성 및 세팅해주는 역할을 한다. NPM같은 패키지 매니저 역할을 한다고 생각하면 쉽다(실제로 NPM을 통해서 동작한다).
3. Vue Router: Vue Router는 가상 DOM 과 Vuex사이의 통로 및 표지판이라고 생각하면 편하다. Vuex상의 컴포넌트를 Router를 통해 다른 곳으로 옮겨 사용할 수 있다.

Vue 인스턴스의 라이프 사이클



출처: <https://jeonghwan-kim.github.io/vue/2017/03/27/vue.html>

* MVVM패턴

View, ViewModel, Model 3가지 요소를 서로 독립적으로 분리시킴으로써 UI와 비즈니스 로직을 분리해서 생산성을 높이고 유지보수를 용이하게 하려는 디지인 패턴.

MVVM 패턴은 [Command 패턴](https://ko.wikipedia.org/wiki/%EC%BB%A4%EB%A7%A8%EB%93%9C_%ED%8C%A8%ED%84%B4)과 [Data Binding](https://en.wikipedia.org/wiki/Data_binding) 두 가지 패턴을 사용하여 구현되어 있다. Command 패턴과 Data Binding을 이용하여 View와 View Model 사이의 의존성을 없앴으며 View Model과 View는 1:n 관계를 가질 수 있다.

View와 Model은 MVC디자인 패턴의 구조와 동일하지만 ViewModel부분에 차이가 존재한다. ViewModel은 Control과 달리 사용자 요청을 처리하는 로직이 있는 것이 아니라 View를 위한 Model 즉 가공된 데이터를 가지고 있다. 이후 View는 화면에 페이지를 출력하기 위해 View Model의 Data를 Binding하여 화면에 출력한다.

* 가상 DOM

기존 DOM(Document Object Model)이 Component를 관리하는 방식 즉 DOM 구조를 모두 체크하고 변경하는 방식의 복잡도와 시간을 줄이기 위해 개발된 방식이다. 기존 DOM방식은 작은 기능 변동이 있더라도 해당 컴포넌트의 위치, 변경된 기능을 모두 서술해 주어야 했고 이는 Rendering 속도를 낮출 뿐 아니라 개발자의 수고도 컸다(정확한 위치를 열거해 줘야 했는데 이는 노력이 많이 들 뿐만 아니라 코드의 직관성도 낮췄다).

가상 DOM은 기존 DOM방식과 달리 Component의 위치와 정보들을 자동 관리해주고 개발자에게는 가상의 위치정보를 사용할 수 있게 지원해 준다. 가상 DOM을 사용하면 Component의 위치 관리뿐 아니라 현재 Component의 상태 조회까지 개발자는 전반적으로 편리함을 얻을 수 있다.

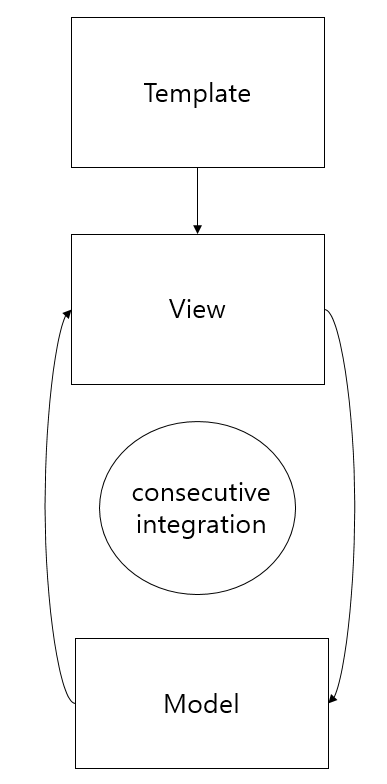
1. Angular JS

Angular JS는 SPA(Single Page Application) 프레임워크다. SPA란 기존 요청시마다 서버에서 리소스 및 데이터를 분석하고 화면에 다시 렌더링 하는 방식과 달리 한 번만 브라우저에 페이지 전체를 로드하고 이후에는 부분적으로 데이터를 바인딩 하는 형식으로 구성되는 방식을 말한다.

HTML은 정적인 마크업 언어로써 요소들을 생성하거나 업데이트 했을 때 UI요소를 Filtering 하는 등 다이나믹 Application에 필요한 다양한 일들을 할 수 없다. 이러한 문제를 해결하기 위한 방법으로 JS를 이용하여 DOM을 수정할 수 있지만 개발자의 능력이 매우 중요하게 된다. AngularJS는 HTML을 확장해서 다이나믹 Application을 지원한다. 예를 들어 <option>태그를 <select>태그에 생성할 때 자바스크립트의 코드의 작성없이 이를 수행한다.

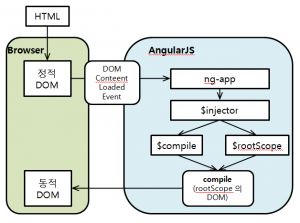
디자인 패턴으로는 MVC 디자인 패턴을 사용하고 있다.

양방향 데이터 결합이라는 특징을 가진다.



위와 같이 뷰는 모델과 양방향 연결이 되어 있어 서로의 변동사항 예를 들어 모델의 데이터에 변동사항이 있으면 뷰는 그에 맞춰 자동으로 업데이트된다.

Angular JS의 부팅과정

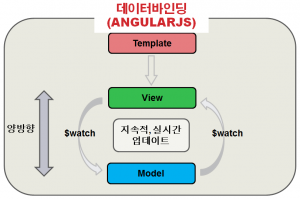


1. 브라우저가 HTML 파일들을 로드 한다.
2. Angular.js 가 로드 된다.
3. DOM Content Loaded Event를 기다린다.
4. DOM이 모두 로드 되면 이벤트를 발생시켜 ng-app 지시자를 찾는다.
5. np-app은 요청이 넘어오면 injector 지시어를 생성한다.
6. injector지시어는 Context(루프 Scope)를 생성하여 하위 DOM들을 컴파일하고 rootScope의 DOM에 연결한다.

Angular JS는 양방향 데이터 바인딩을 지원한다.

$watch명령어를 이용하여 모델과 뷰가 서로를 실시한으로 모니터링한다. 뷰나 모델에 변경이 발생하면 변경을 감지하여 상태를 변경한다. 예를 들어 뷰의 Input Box에 변경이 생기면 모델은 변경이 발생했음을 뷰에 알리고 뷰는 변경사항을 현재 표시중인 화면에 적용한다.

이러한 방식은 Angular JS가 동일한 이름의 input과 데이터 모델을 매핑해 줌으로써 구현된다. Angular JS는 이 둘을 매핑하고 둘의 싱크가 유지되도록 해주는 역할을 한다. 개발자는 자바스크립트 속성을 할당할 UI만 선택하여 적용하면 자동으로 동기화가 이루어진다.



위는 이를 도식화한 그림이다.

----------------------------------------------------------------------------------------------------

**참고사항**

**스프링 프레임워크 특징 간단정리**

* 경량컨테이너로 라이프사이클을 관리하고 필요한 객체를 스프링으로부터 받아온다.
* DI지원하여 객체 간의 의존관계 설정이 가능하다.
* AOP지원한다.
* POJO방식으로 자바객체는 특정한 인터페이스를 구현하고 클래스 상속이 필요치 않다.
* Transaction처리를 위한 일관된 방법을 제공한다.
* 영속성 관련 다양한 API를 지원합니다. API연동을 지원한다.

**언어별 프레임워크 정리**

* 자바 프레임워크 – Spring, 전자정부 프레임워크 등
* 자바스크립트 프레임워크 – AngularJS, React 등
* PHP 프레임워크 – Laravel, Symfony 등

**Persistent 프레임워크**

* Persistent 프레임워크란?

데이터베이스와의 연동되는 시스템을 빠르게 개발하고 안정적인 구동을 보장해주는 프레임워크. 자료를 데이터베이스에 저장하는 과정을 지원하며 어플리케이션의 데이터(or 객체)와 데이터 베이스 안의 실제 데이터 사이의 간극을 줄일 수 있다.

데이터의 저장, 조회, 변경, 삭제를 다루는 클래스와 설정파일의 집합으로 이루어 지는데 이를 이용하면 복잡한 프로그래밍 없이 데이터베이스와 연동할 수 있고 이를 자동화할 수도 있다.

* Persistent 프레임워크의 종류
  + - * ORM

Object Relation Mapping의 약자 그 뜻 그대로 객체와 관계형 데이터베이스의 데이터 매핑해서 관계형 데이터베이스의 테이블을 객체처럼 사용할 수 있게 해주는 방식.

가장 큰 특징은 SQL문 없이 일반 코드를 통해서 객체와 테이블의 매핑 설정 만으로 테이블을 객체에 담을 수 있다는 것이다.

ORM을 사용하는 Persistent 프레임워크는 Hibernate가 있다. Hibernate는 SQL대신 메소드를 이용해서 데이터에 접근하는 방식을 사용한다(내부적으로는 SQL이 사용되기 때문에 SQL이 아예 사용되지 않는 것은 아니다). 이러한 방식은 개발자가 쿼리보다 비즈니스 로직 개발에 집중하게 해주며 객체지향 개발에 유리하다. 그러나 성능 부분이나 SQL문을 직접 조작하기 힘들다는 단점 또한 존재한다.

* + - * SQL Mapper

객체와 SQL문을 매핑해서 관계형 데이터를 객체화 하는 방식. SQL질의 결과와 객체를 매핑해 준다는 점에서 ORM방식과 차이가 있다. SQL질의 결과와 객체를 매핑하기 때문에 개발자가 직접 쿼리를 작성하고 관리해야 한다.

SQL Mapper를 사용하는 Persistent 프레임워크는 MyBATIS/iBATIS(이하 MyBATIS)가 있다. MyBATIS는 JDBC 관련 설정과 SQL결과 매핑을 개발자 대신 관리해준다. 예를 들면 Connection등 DB관련 여러 설정들을 개발자가 직접 다루어야 했으나 MyBATIS에서는 속성 파일에 이런 연결 정보를 등록해 두면 알아서 관리해 준다. 이러한 방식은 개발자가 SQL전반을 관리할 수 있다는 점과 어플리케이션에 최적화된 쿼리문을 사용할 수 있다는 장점을 가진다. 또한 어떤 시스템에서도 평균적인 성능이 나온다. 반면 객체와 쿼리문을 개발자가 직접 관리해야 하기 때문에 생산성이 떨어질 수 있고 데이터 베이스에 대한 종속성이 생긴다(DB가 변경될 경우 함수들도 변경해 주어야 함.

**테스트 자동화 프레임워크**

테스트 자동화를 위해서는 스크립트가 필요하다. 프로젝트 규모가 커질수록 이런 스크립트를 작성하고 유지보수 하는 일에 부담이 커지게 되는데 테스트 자동화 프레임워크는 이러한 작업을 수월하게 처리할 수 있도록 지원한다. 테스트 자동화 프레임 워크가 없다면 어플리케이션의 작은 변화에도 테스트 스크립트는 수십 혹은 수백개의 수정이 필요할 수 있다.

테스트 자동화 프레임워크에는 5가지 종류가 존재한다

* 선형 프레임워크
* 모듈식 프레임워크
* 데이터 주도 프레임워크
* 키워드 주도 프레임워크

추가 SVN, ANT