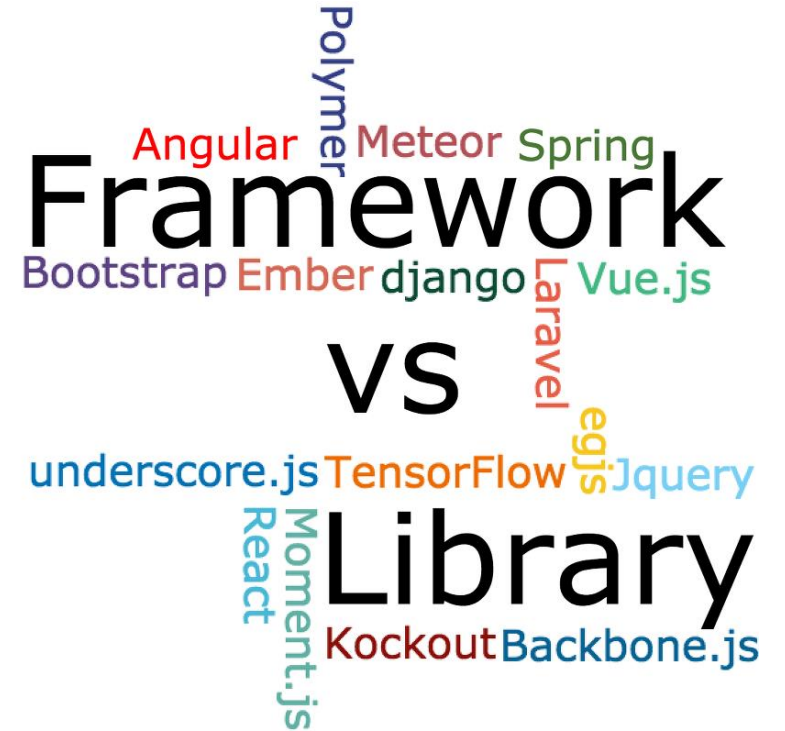


Framework 란?

- ▶ “잘 고른 프레임워크, 프로젝트 성패를 가른다.”
- ▶ 프레임워크 vs 라이브러리
 - ▶ 프레임워크: 코드의 품질, 필수적인 코드, 알고리즘, 보안, 데이터베이스 연동 같은 기능들을 어느정도 구성이 되어있는 뼈대(구조)를 제공
 - ▶ 라이브러리: 라이브러리는 특정 기능에 대한 API(도구 / 함수)를 모은 집합
- ▶ 프레임워크 선정 기준 예시
 - ▶ 재사용성이 좋을 것
 - ▶ 확장성이 좋을 것
 - ▶ 사용하기 쉬울 것
 - ▶ 플랫폼으로 부터의 독립성
 - ▶ 꾸준한 지원



Spring Framework 란?

▶ Spring Framework 란?

- ▶ 2002년 로드 존슨 저술(Expert One-on-One J2EE Design and Development)
- ▶ 2004년 1.0, ..., 2013년 4.0, 2017년 5.0, 최종 5.3.9
- ▶ 자바 플랫폼을 위한 오픈소스 애플리케이션 프레임워크로서 동적인 웹 사이트를 개발하기 위한 여러 가지 서비스를 제공
- ▶ Spring 관련 프로젝트
 - ▶ Spring MVC : 웹 애플리케이션 개발을 위한 프레임워크
 - ▶ Spring 시큐리티 : 애플리케이션에 인증이나 인가 같은 보안 기능을 쉽게 구현할 수 있도록 도와주는 프레임워크
 - ▶ Spring 데이터 : NoSQL, Key-Value 등 다양한 데이터 저장소의 데이터에 쉽게 접근할 수 있게 해주는 프레임워크
 - ▶ Spring 배치 : 스프링 배치 애플리케이션을 개발하기 위한 경량 프레임워크
 - ▶ Spring 인티그레이션 : 시스템 연계(AMQP, 파일, FTP, REST, JDBC, JPA, ...) 애플리케이션을 쉽게 개발할 수 있게 도와주는 프레임워크
 - ▶ Spring 클라우드 : 클라우드 환경에 최적화된 애플리케이션 개발을 위한 프레임워크 (Cloud Config, Cloud Netflix, Cloud Bus, Cloud Connectors 등)
 - ▶ Spring 툴 스위트(STS) : 스프링 기반 애플리케이션을 개발하기에 최적화된 이클립스 기반 통합 개발환경
 - ▶ Spring IO 플랫폼 : 스프링 관련 라이브러리나 서드파티 라이브러리의 버전을 결정하고 의존 관계를 해결하기 위한 프로젝트
 - ▶ Spring Boot : 최소한의 설정만으로 프로덕션 레벨의 스프링 기반 애플리케이션을 쉽게 개발할 수 있게 도와주는 프로젝트
 - ▶ ...

▶ 전자정부 표준 프레임워크

- ▶ 공공사업에 적용되는 개발 프레임워크의 표준 정립으로 응용 SW 표준화, 품질 및 재 사용성 향상을 목표로 함.
- ▶ 2009년 1.0, 2021년 3.10.0(Spring Framework 4.3.25 기반)
- ▶ <https://www.egovframe.go.kr/>

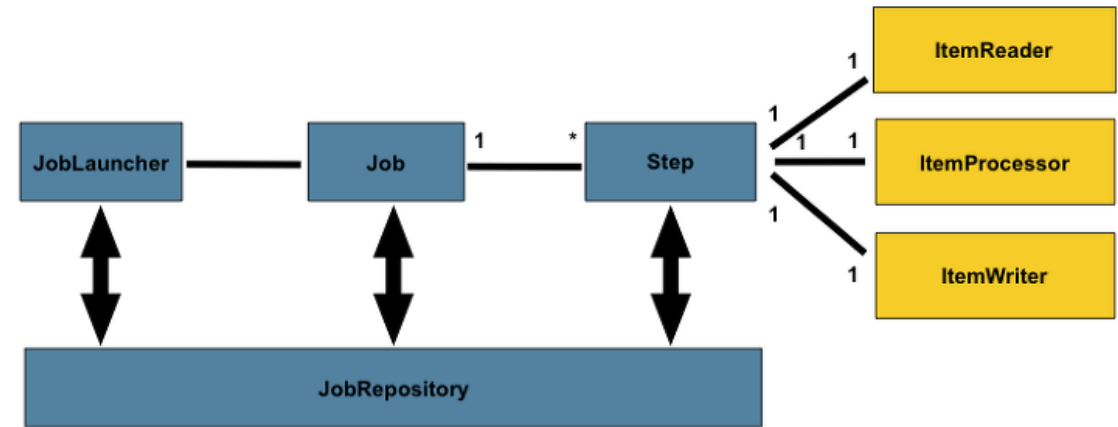
Spring Boot 소개

▶ Spring Boot란?

- ▶ 2014년 1.0, 현재 2.5
- ▶ 단독으로 실행이 가능한 스프링 애플리케이션을 생성함
- ▶ 스프링 부트는 스프링을 쉽게 사용할 수 있도록 필요한 설정을 대부분 미리 세팅 해 놓았음
- ▶ 가능한 자동으로 설정되어 있음
- ▶ 상용화에 필요한 통계, 상태 체크, 외부 설정 등을 제공
- ▶ 설정을 위한 XML 코드를 생성하거나 요구하지 않음
- ▶ Tomcat, Jetty, Undertow 웹서버 내장
- ▶ Jar, War 파일로 배포 가능

Spring Boot 배치

- ▶ 백엔드 단의 배치 처리 프로그램
- ▶ 대용량 데이터 처리에 최적화되어 고성능
- ▶ 효과적인 로깅, 통계처리, 트랜잭션 관리
- ▶ 수동으로 처리하지 않도록 자동화
- ▶ 예외사항과 비정상 동작에 대한 방어 기능
- ▶ 멀티 스레드 적용 종류
 - ▶ TaskExecutor 사용하여 여러 Step 동작
 - ▶ 여러 개의 Flow 실행
 - ▶ 파티셔닝을 사용한 병렬 프로그래밍
- ▶ 배치 수행 시 배치관리 테이블 생성 필요함
 - ▶ 배치 실행 상태 관리
 - ▶ 스케줄러 연동 가능



폴더 구성

- ▶ 프로젝트의 크기에 따라 미리 계획하여 관리
- ▶ 파일의 양이 많을 경우 계층 구조로 관리
- ▶ static은 외부에서 접근 가능한 파일 관리

```
demo [boot]
├── src/main/java
│   └── com.example.demo
│       ├── DemoApplication.java
│       ├── HelloController.java
│       └── ServletInitializer.java
├── src/main/resources
│   ├── templates
│   ├── static
│   └── application.properties
├── src/test/java
├── JRE System Library [JavaSE-1.8]
├── Project and External Dependencies
├── bin
├── gradle
├── src
│   ├── build.gradle
│   ├── gradlew
│   ├── gradlew.bat
│   ├── HELP.md
│   └── settings.gradle
```

확장성 있는 구조로 변경



```
qhedge [boot] [devtools] [qhedge master]
├── src/main/java
│   └── com.qhedge
│       ├── aop
│       ├── batch.stocktrading
│       ├── utils
│       ├── StockTradingSimulator.java
│       ├── StockTradingSimulatorConfig.java
│       ├── StockTradingSimulatorDecider.java
│       ├── common
│       ├── configuration
│       ├── interceptor
│       ├── online
│       ├── board
│       ├── login.controller
│       └── stock
│           ├── controller
│           │   ├── DashBoardController.java
│           │   ├── PatternController.java
│           │   └── StatisticsController.java
│           ├── dto
│           │   ├── JongmokDto.java
│           │   ├── PatternDetailDto.java
│           │   ├── PatternMasterDto.java
│           │   ├── PatternPoolDto.java
│           │   ├── PatternRevenueDailyDto.java
│           │   ├── PatternRevenueDto.java
│           │   ├── PatternRevenueMinutelyDto.java
│           │   ├── StockDailyCountDto.java
│           │   ├── StockDailyDto.java
│           │   ├── StockDailyMinuteDto.java
│           │   └── TradingPolicyDto.java
│           ├── mapper
│           │   ├── PatternMapper.java
│           │   ├── StatisticsMapper.java
│           │   ├── StockMapper.java
│           │   └── TradingMapper.java
│           └── service
│               ├── DashBoardService.java
│               ├── DashBoardServiceImpl.java
│               ├── PatternService.java
│               ├── PatternServiceImpl.java
│               ├── StatisticsService.java
│               └── StatisticsServiceImpl.java
├── security
├── QhedgeApplication.java
├── ServletInitializer.java
├── src/test/java
```

```
src/main/resources
├── mapper
│   ├── sql-board.xml
│   ├── sql-pattern.xml
│   ├── sql-statistics.xml
│   ├── sql-stock.xml
│   └── sql-trading.xml
├── templates
│   ├── board
│   ├── common
│   ├── layout
│   │   └── comm_layout.html
│   ├── login
│   └── stock
│       ├── blank.html
│       ├── charts.html
│       ├── main.html
│       ├── patternDailyChart.html
│       ├── patternDailyChart2.html
│       ├── patternMinutelyChart.html
│       ├── stockDailyChart.html
│       ├── stockDailyMinutelyChart.html
│       ├── stockMinutelyChart.html
│       ├── stockMinutelyChart2.html
│       └── tables.html
├── gulpfile.js
├── package-lock.json
├── package.json
├── static
│   ├── application.properties
│   ├── log4jdbc.log4j2.properties
│   └── logback-spring.xml
```

Annotation

▶ Annotation이란?

- ▶ @를 이용한 주석, 자바코드에 주석을 달아 특별한 의미를 부여한 것 (참고로 클래스, 메소드, 변수 등 모든 요소에 선언이 가능)
- ▶ 메타데이터(실제데이터가 아닌 Data를 위한 데이터) 라고도 불리고 JDK5부터 등장
- ▶ 컴파일러가 특정 오류를 억제하도록 지시하는 것과 같이 프로그램 코드의 일부가 아닌 프로그램에 관한 데이터를 제공, 코드에 정보를 추가하는 정형화된 방법.

ex) @Override, @SuppressWarnings

@Repository, @Service, @Controller, @Autowired, @Resource

@Slf4j, @EnableBatchProcessing, @SpringBootApplication, @RestController, @RequestMapping

▶ Annotation이 나온 이유

- ▶ IT가 발전하면서 프로그램의 규모가 방대해지면서 XML이 가지는 설정정보의 양이 많아 짐
→ Annotation은 직관적인 메타데이터 설정이 가능. 왜냐하면 소스코드와 같이 쓰기 때문에 (소스코드와 메타데이터가 결합되는 형태)

▶ Annotation 사용시 장점

- ▶ 데이터에 대한 유효성 검사조건을 **Annotation** 을 사용하여 Model 클래스에 직접 명시함으로써 해당 데이터들에 대한 유효 조건을 쉽게 파악할 수 있게 되며, 코드의 양도 줄어 듦

JQuery

- ▶ <https://jquery.com/>
- ▶ jQuery는 자바스크립트의 생산성을 향상시켜주는 자바스크립트 라이브러리
- ▶ jQuery는 오늘날 가장 인기있는 자바스크립트 라이브러리 중 하나
- ▶ DOM(Document Object Model) 요소 선택 기의 파생 프로젝트
- ▶ AJAX(Asynchronous JavaScript and XML) 함수 제공
 - ▶ MVC 모델의 단점 개선(화면을 리프레쉬 하지 않고 데이터 조회)
- ▶ JSon 파싱

```
$.ajax({  
  url: "/pattern/patternRevenu.do",  
  type: "GET",  
  data: {'patternId':id, 'openDate':openDate,  
        'closeDate':closeDate},  
  success: function(data){  
    drawChart(data);  
  },  
  error: function() {  
    alert("조회중 오류가 발생하였습니다.");  
  }  
});
```

```
document.getElementById('retrieve').addEventListener('click', function() {  
  ...  
});
```

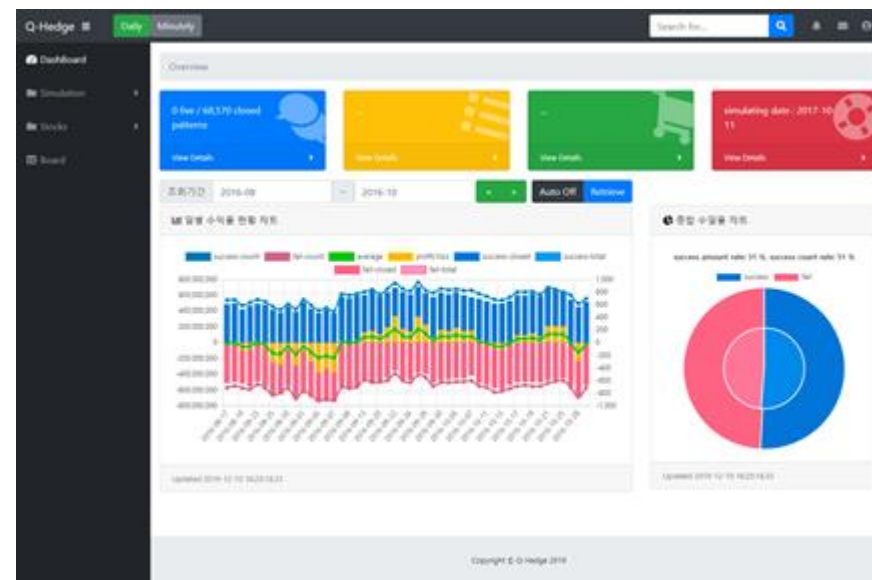


```
$('#retrieve').on('click', function() {  
  ...  
});
```

Thymeleaf

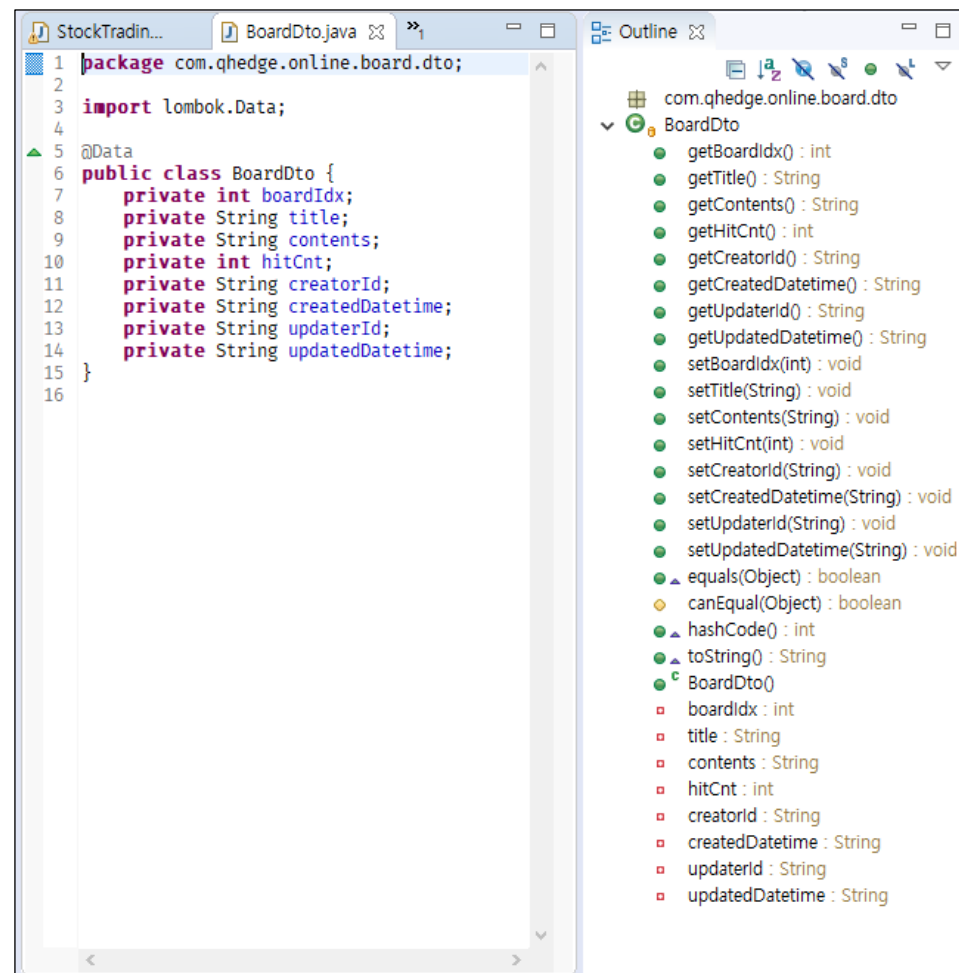
- ▶ <https://www.thymeleaf.org/>
- ▶ HTML, XML, JavaScript, CSS 및 일반 텍스트를 처리 할 수 있는 Java 템플릿 엔진
- ▶ Thymeleaf는 html파일을 가져와서 파싱해서 분석 후 정해진 위치에 데이터를 치환해서 웹 페이지를 생성
- ▶ JSP(Java Server Pages) 파일에 비즈니스 로직을 넣으면 디버깅 및 유지보수가 힘들어져 요즘은 JSP에서는 자바 코드를 사용하지 못하게 하는게 일반적
- ▶ Layout 기능 제공(독립적 구동 가능 형태로 존재)
- ▶ 성능 : Freemarker > Velocity > JSP > Thymeleaf

```
<table>
<tr>
<th>NAME</th>
<th>PRICE</th>
<th>IN STOCK</th>
</tr>
<tr th:each="prod : ${prods}">
<td th:text="${prod.name}">Onions</td>
<td th:text="${prod.price}">2.41</td>
<td th:text="${prod.inStock}? #{true} : #{false}">yes</td>
</tr>
</table>
```



Lombok

- ▶ <https://projectlombok.org/>
- ▶ VO(Value Object): 사용 되는 값이 객체로 표현
- ▶ DTO(Data Transfer Object): 데이터의 전송을 위한 객체이며, 비즈니스 로직까지 담아서 사용
- ▶ Annotation 설정으로 Setter/Getter, ToString, equals, hashCode 함수 자동생성
 - ▶ @Data, @Setter, @Getter, @ToString



```
1 package com.qhedge.online.board.dto;
2
3 import lombok.Data;
4
5 @Data
6 public class BoardDto {
7     private int boardIdx;
8     private String title;
9     private String contents;
10    private int hitCnt;
11    private String creatorId;
12    private String createdDatetime;
13    private String updaterId;
14    private String updatedDatetime;
15 }
16
```

Outline view for BoardDto:

- getBoardIdx() : int
- getTitle() : String
- getContents() : String
- getHitCnt() : int
- getCreatorId() : String
- getCreatedDatetime() : String
- getUpdaterId() : String
- getUpdatedDatetime() : String
- setBoardIdx(int) : void
- setTitle(String) : void
- setContents(String) : void
- setHitCnt(int) : void
- setCreatorId(String) : void
- setCreatedDatetime(String) : void
- setUpdaterId(String) : void
- setUpdatedDatetime(String) : void
- equals(Object) : boolean
- canEqual(Object) : boolean
- hashCode() : int
- toString() : String

BoardDto class fields:

- boardIdx : int
- title : String
- contents : String
- hitCnt : int
- creatorId : String
- createdDatetime : String
- updaterId : String
- updatedDatetime : String

Bootstrap

- ▶ <https://getbootstrap.com/>
- ▶ 서로 다른 인터페이스를 사용한 여러 개발자들의 공동작업
- ▶ 디자인 불일치, 관리 어려움, 일관성 유지 불가
- ▶ 문제점 개선을 위해 트위터 개발자와 UI 디자이너가 개발
- ▶ 프론트엔드 개발을 빠르고 쉽게 할 수 있는 프레임 워크
- ▶ HTML과 CSS 기반의 템플릿 양식, 버튼, 네비게이션 및 기타 페이지를 구성하는 요소 포함

