



Spring Boot

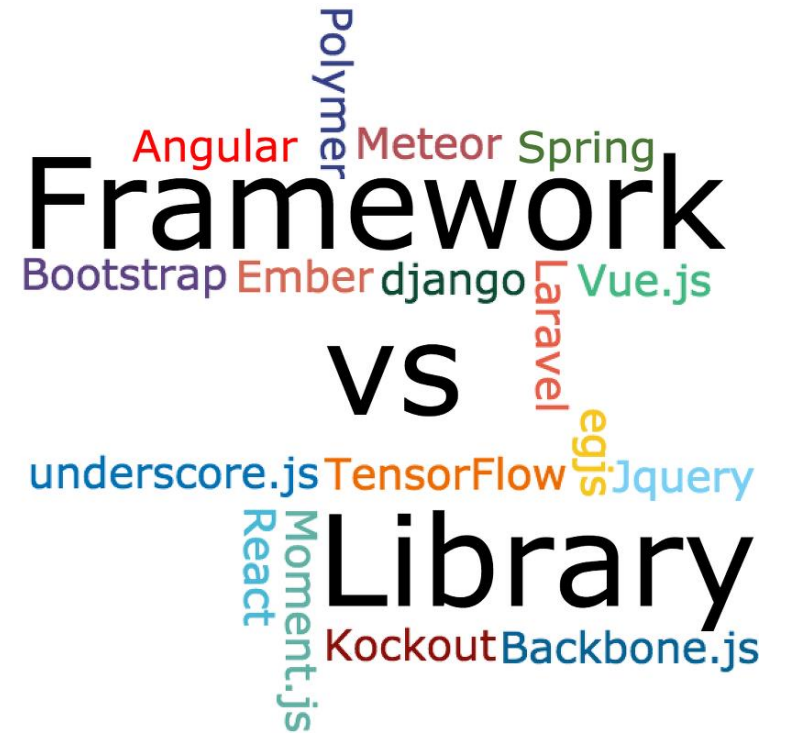
소프트웨어융합공학과
웹서버 프로그래밍
허우행

목차

- ▶ Framework 란?
- ▶ Spring Framework 란?
- ▶ Spring Boot 소개
- ▶ Spring Boot 환경설정
- ▶ Spring Boot 프로젝트 생성
- ▶ Spring Boot 프로젝트 실행
- ▶ Spring Boot 빌드 및 배포
- ▶ Spring Boot 웹(MVC)
- ▶ Spring Boot 배치
- ▶ Spring Boot 폴더 구성
- ▶ Spring Boot 의 다양한 구성

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/>

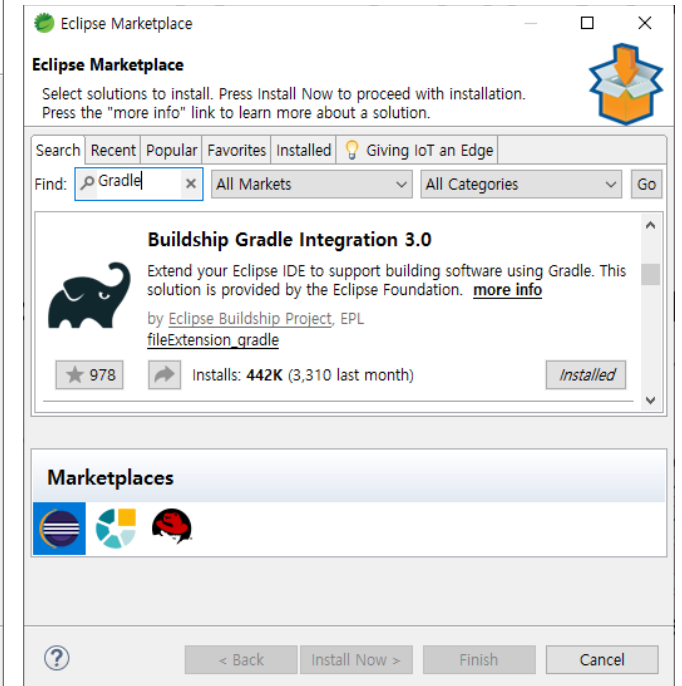
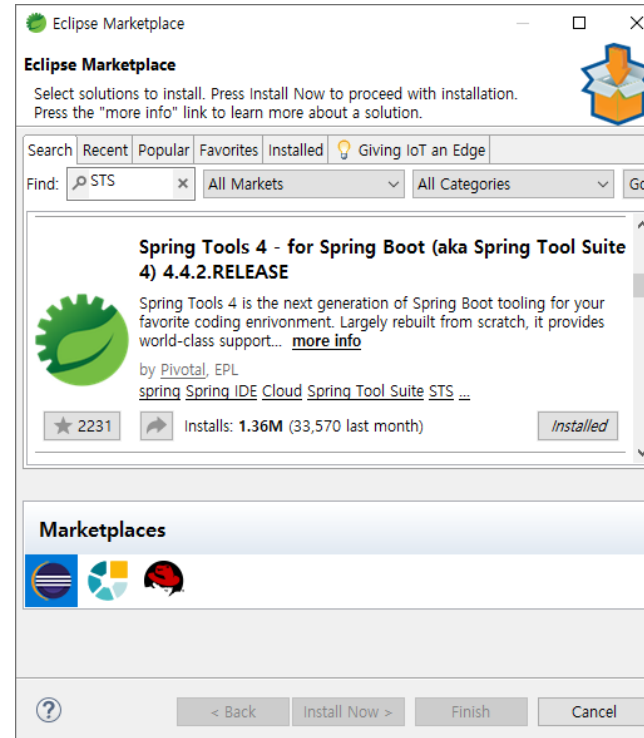
Spring Boot 소개

▶ Spring Boot 란?

- ▶ 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 환경설정

- ▶ JDK 설치(SE 8 vs SE 16)
 - ▶ <https://www.oracle.com/technetwork/java/javase/downloads/index.html>
- ▶ Eclipse or IntelliJ(유료) 설치
 - ▶ <https://www.eclipse.org/downloads/>
 - ▶ <https://spring.io/tools> (spring-tool-suite-4-4.3.2 포함)
- ▶ STS(Spring Tool Suite) 플러그인 설치
 - ▶ Eclipse 의 Help → Eclipse Marketplace
- ▶ Gradle or Maven 설치
 - ▶ Eclipse 의 Help → Eclipse Marketplace
- ▶ 플러그인 설치
 - ▶ Lombok(직접 설치 후 사용가능)
<https://projectlombok.org/download>
 - ▶ Jad(Java 디컴파일러)
 - ▶ ...





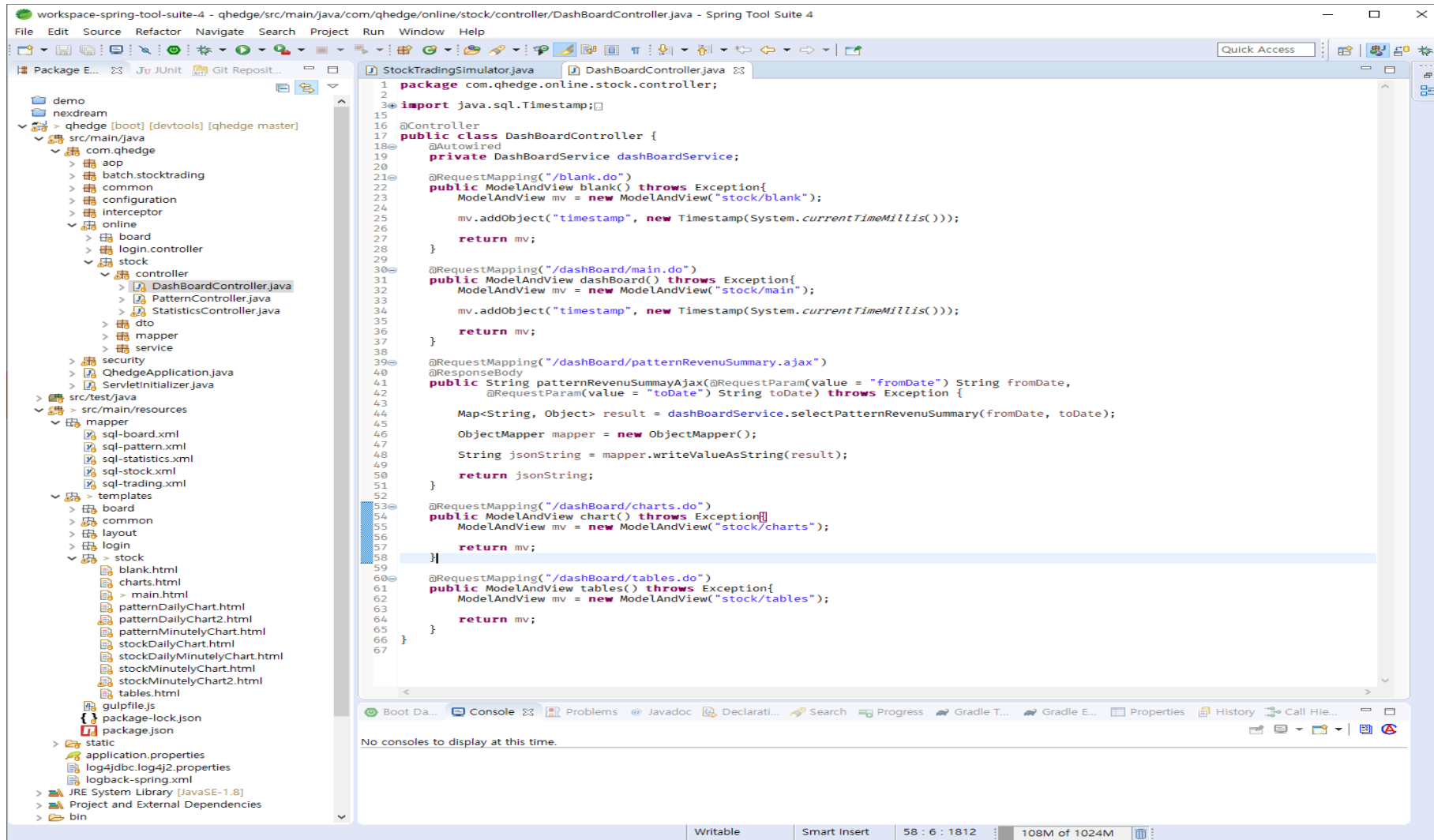
데이터베이스 설정하기



- ▶ MySQL or MariaDB
 - ▶ <https://www.mysql.com/downloads/> - MySQL DB
 - ▶ <https://downloads.mariadb.org/> - Maria DB ← HeidiSQL이 같이 설치됨.
- ▶ HeidiSQL or Workbench
 - ▶ <https://dev.mysql.com/downloads/workbench/>
 - ▶ <https://www.heidisql.com/download.php>



Eclipse STS4



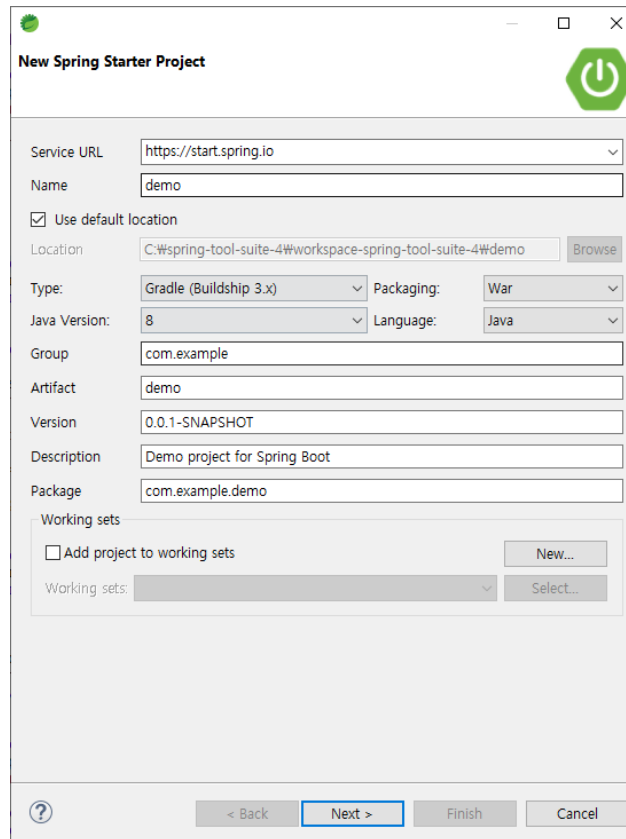
Spring Boot 프로젝트 생성

▶ 프로젝트 생성

- ▶ File → New → Spring Starter Project
- ▶ Gradle or Maven 선택
- ▶ Jar(Java ARchives) or War(Web application Archives) 선택
- ▶ Java or Kotlin or Groovy 선택
- ▶ 의존성 추가
 - ▶ Web, H2(Jdbc 설정 안할 경우 필요), DevTools, Lombok, MyBatis, Security, ...

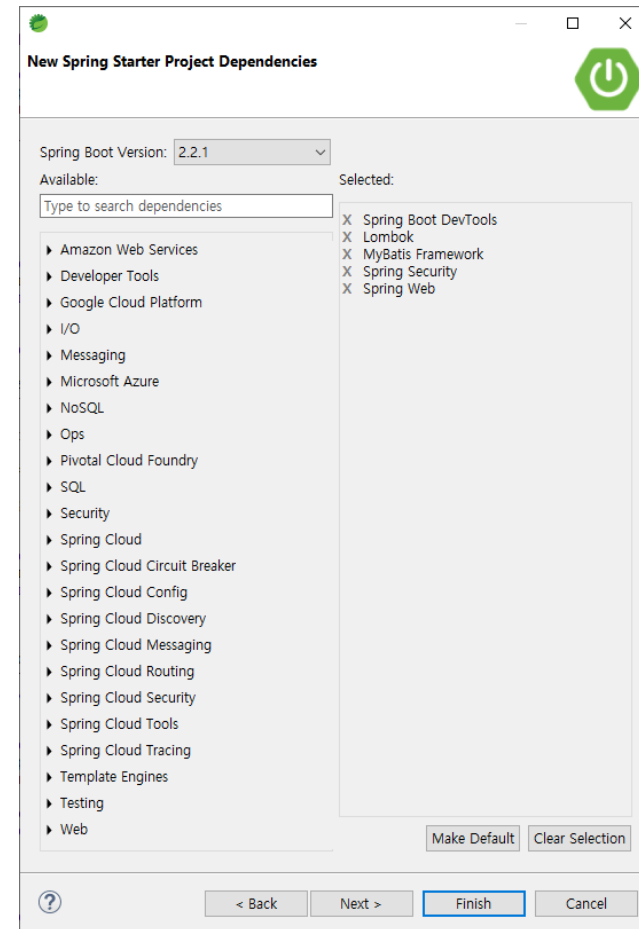
build.gradle

```
plugins {  
    id 'org.springframework.boot' version '2.2.1.RELEASE'  
    id 'io.spring.dependency-management' version '1.0.8.RELEASE'  
    id 'java'  
    id 'war'  
}  
...  
dependencies {  
    implementation 'org.springframework.boot:spring-boot-starter-web'  
    runtimeOnly 'com.h2database:h2'  
    providedRuntime 'org.springframework.boot:spring-boot-starter-tomcat'  
    testImplementation('org.springframework.boot:spring-boot-starter-test') {  
        exclude group: 'org.junit.vintage', module: 'junit-vintage-engine'  
    }  
}
```



The 'New Spring Starter Project' dialog box is shown. It contains the following fields and options:

- Service URL: `https://start.spring.io`
- Name: `demo`
- ☒ Use default location
- Location: `C:\spring-tool-suite-4\workspace-spring-tool-suite-4\demo`
- Type: `Gradle (Buildship 3.x)`
- Packaging: `War`
- Java Version: `8`
- Language: `Java`
- Group: `com.example`
- Artifact: `demo`
- Version: `0.0.1-SNAPSHOT`
- Description: `Demo project for Spring Boot`
- Package: `com.example.demo`
- Working sets: ☐ Add project to working sets



The 'New Spring Starter Project Dependencies' dialog box is shown. It contains the following fields and options:

- Spring Boot Version: `2.2.1`
- Available: `Type to search dependencies`
- Selected:
 - ☒ Spring Boot DevTools
 - ☒ Lombok
 - ☒ MyBatis Framework
 - ☒ Spring Security
 - ☒ Spring Web

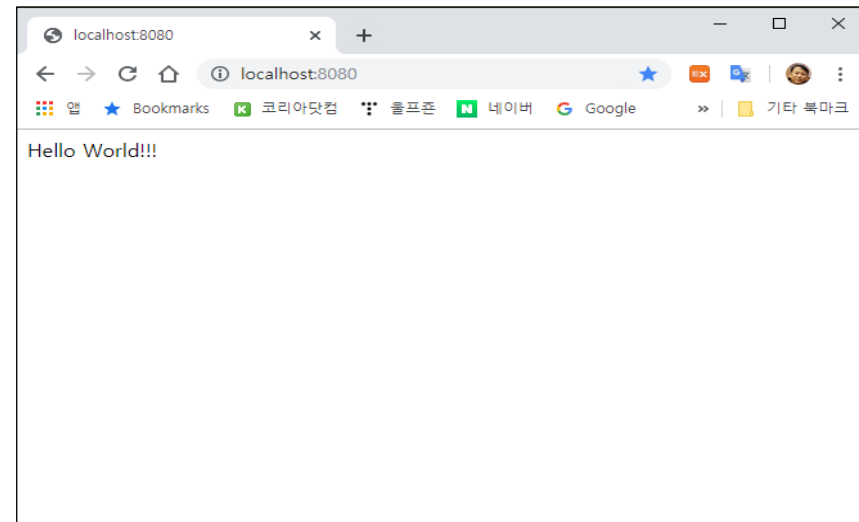
Spring Boot 프로젝트 실행

- ▶ 의존성 설정에 대한 Jar 다운로드(의존성 변경시마다 작업필요)
 - ▶ 프로젝트 선택 후 오른쪽 마우스 → Gradle → Refresh Gradle Project
- ▶ HelloController.java 생성
- ▶ 웹 서버 실행
 - ▶ 프로젝트 선택 후 오른쪽 마우스 → Run As → Spring Boot App
- ▶ DevTools 의존성 추가 시 소스 변경내용이 즉시 반영됨(미 추가시는 재 기동 해야 반영됨)

```
package com.example.demo;
```

Import...

```
@RestController
public class HelloController {
    @RequestMapping("/")
    public String hello() {
        return "Hello World!!!";
    }
}
```





▶ Build 방법(2가지)

- ▶ Eclipse - Gradle Tasks 창에서 demo\build\build 선택 후 Run Gradle Tasks 메뉴 실행
- ▶ 도스창 - 프로젝트 폴더로 이동 후 → demo>gradlew.bat build
- ▶ 빌드되면 프로젝트 폴더 demo\build\libs 에 demo-o.0.1-SNAPSHOT.war 단일 파일로 생성됨

- ▶ Windows 나 Linux에 Jre 설치하면 실행가능

- ▶ `java -jar demo-0.0.1-SNAPSHOT.war`

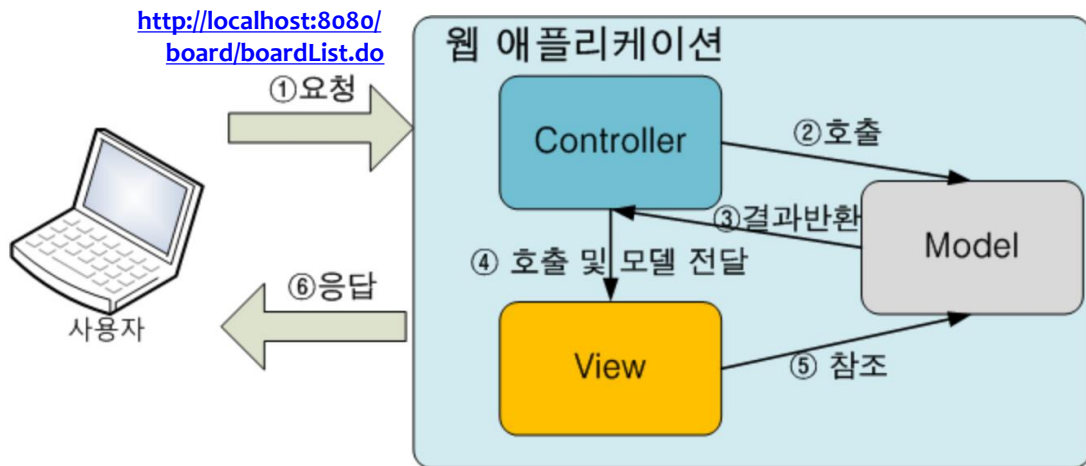
[illegible]

```
2019-11-20 15:51:42,511 INFO [com.demo.DemoApplication] Starting DemoApplication on DESKTOP-30ED31T with PID 15376
(C:\Users\woohaeng\git\demo\build\libs\demo-0.0.1-SNAPSHOT.war started by woohaeng in C:\Users\woohaeng\git\demo\build\libs)
2019-11-20 15:51:42,512 INFO [com.demo.DemoApplication] No active profile set, falling back to default profiles: default
2019-11-20 15:51:43,757 DEBUG [jdbc.sqltiming] com.zaxxer.hikari.pool.PoolBase.executeSql(PoolBase.java:567)
1. SELECT 1 {executed in 9 msec}
HikariDataSource (HikariPool-1)
```

Spring Boot 웹(MVC)

▶ MVC 모델

- ▶ Model : 어플리케이션의 정보, 즉 데이터(DB)
- ▶ View : 사용자에게 보여질 화면(html)
- ▶ Controller : 모델과 뷰의 중계역할(html과 데이터 결합)



```
package com.example.demo;
```

```
import ...
```

```
@RestController
```

```
public class HelloController {
```

```
    @RequestMapping("/board/boardList.do")
```

```
    public ModelAndView openBoardList () throws Exception {  
        ModelAndView mv = new ModelAndView("board/boardList");
```

```
        List<BoardDto> list = boardService.selectBoardList();  
        mv.addObject("list", list);
```

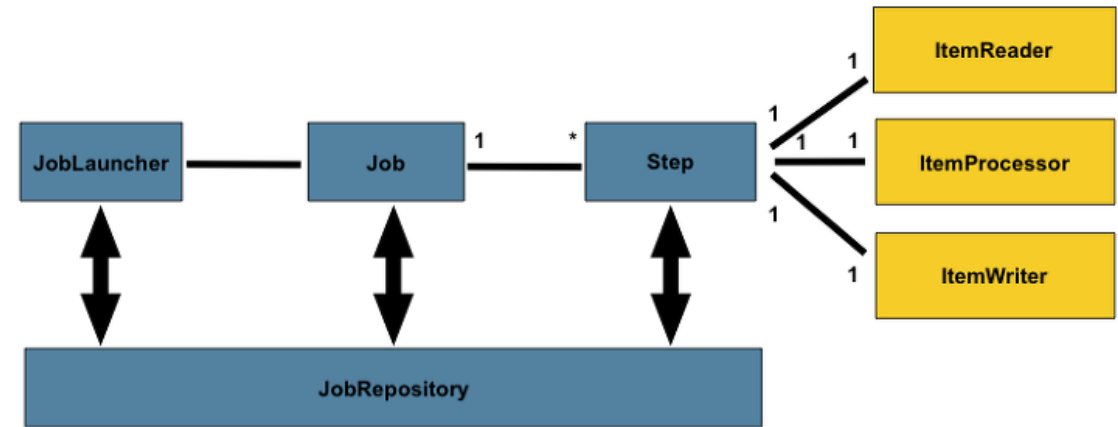
```
        return mv;
```

```
    }
```

```
}
```

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() {  
  ...  
});
```

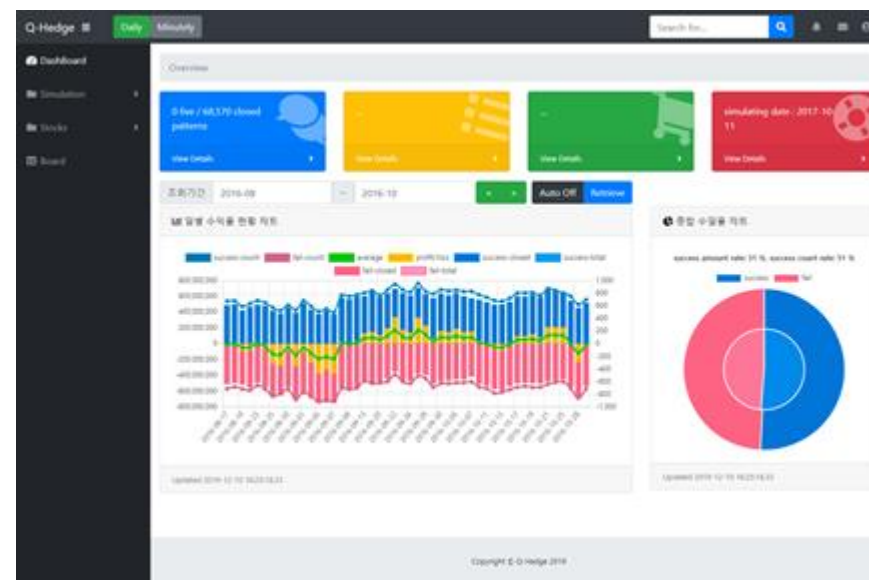


```
$('#retrieve').on('click', function() {  
  ...  
});
```


Thymeleaf

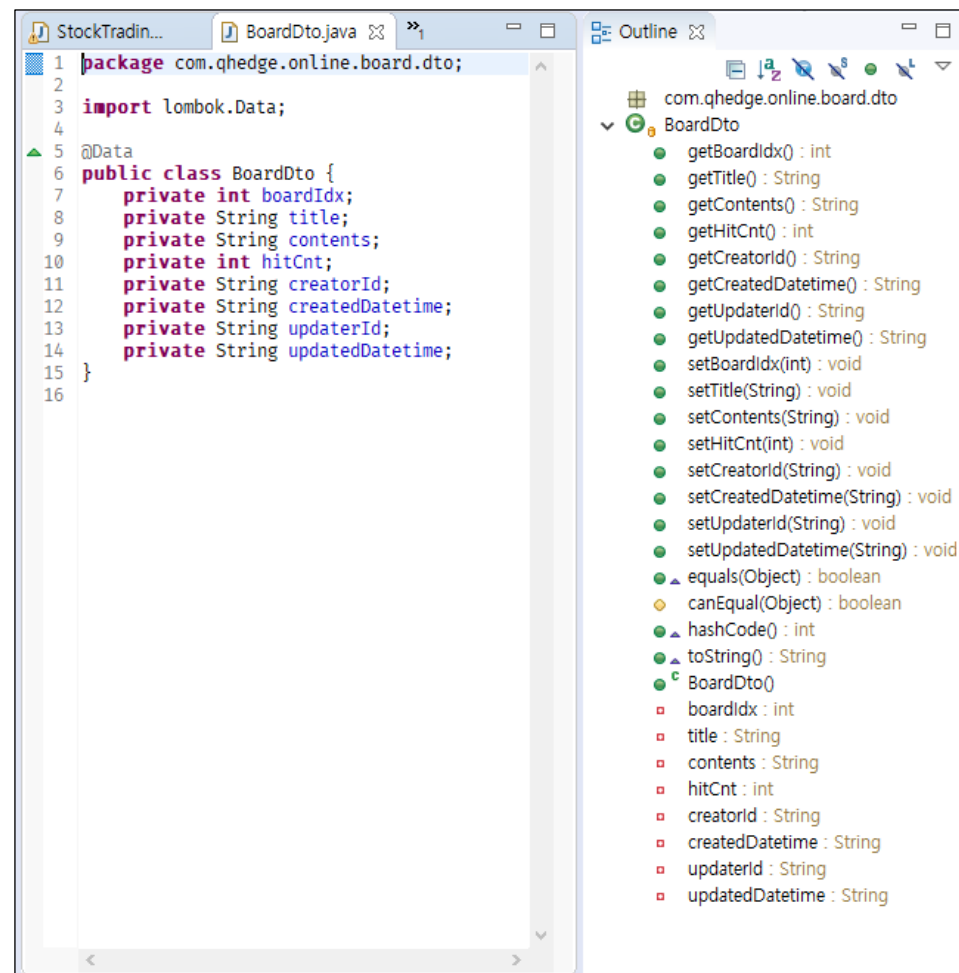
- ▶ <https://www.thymeleaf.org/>
- ▶ HTML, XML, JavaScript, CSS 및 일반 텍스트를 처리 할 수 있는 Java 템플릿 엔진
- ▶ Thymeleaf는 html파일을 가져와서 파싱해서 분석 후 정해진 위치에 데이터를 치환해서 웹 페이지를 생성
- ▶ JSP(Java Server Pages) 파일에 비즈니스 로직을 넣으면 디버깅 및 유지보수가 힘들어져 요즘은 JSP에서는 자바 코드를 사용하지 못하게 하는게 일반적
- ▶ Layout 기능 제공(독립적 구동 가능 형태로 존재)
- ▶ 성능 :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



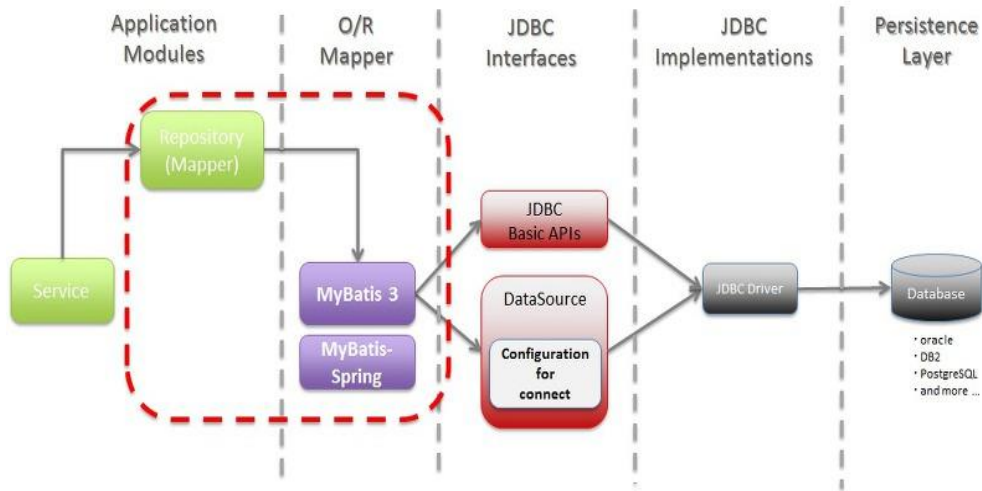
Lombok

- ▶ <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



MyBatis

- ▶ <https://blog.mybatis.org/>
- ▶ 프로그램 코드와 SQL 쿼리의 분리(XML or JAVA) 로 코드의 간결성 및 유지보수성 향상
- ▶ 복잡한 쿼리나 다이내믹 쿼리 지원
- ▶ 조회결과를 사용자 정의 DTO, MAP 등으로 맵핑
- ▶ 빠른 개발이 가능하여 생산성 향상



```
<select id="selectPatternMasterList" resultType="com.qhedge.online.stock.dto.PatternMasterDto">
  SELECT pattern_id,
  account_id,
  DATE_FORMAT(creation_date, '%Y-%m-%d') creation_date,
  pattern_type
  FROM pattern_master
  <where>
    <if test="fromDate != null">
      and open_date >= STR_TO_DATE(CONCAT(LEFT('${fromDate}', 7), '-01'), '%Y-%m-%d')
    </if>
    <if test="toDate != null">
      and open_date <= LAST_DAY(STR_TO_DATE(CONCAT(LEFT('${toDate}', 7), '-01'), '%Y-%m-%d'))
    </if>
  </where>
</select>
```

Logback

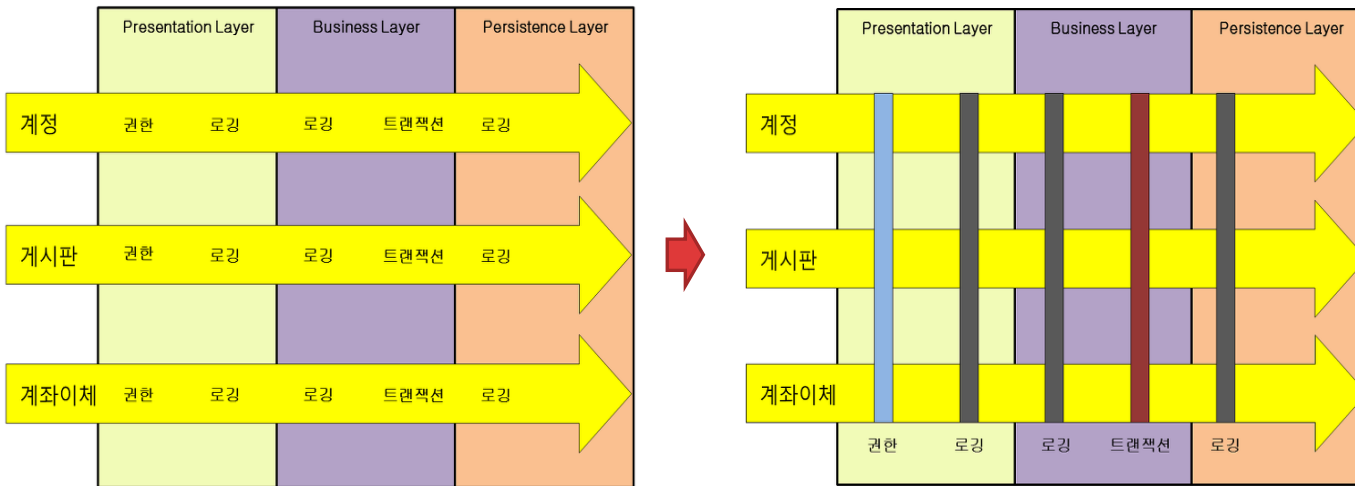
- ▶ <http://logback.qos.ch/>
- ▶ log4j보다 약 10배 정도 빠르게 수행되도록 내부가 변경되었으며, 메모리 효율성 향상
- ▶ 설정 파일을 변경하였을 경우, 서버 재기동 없이 변경 내용이 자동으로 갱신
- ▶ RollingFileAppender를 사용할 경우 자동적으로 오래된 로그를 지워주며 Rolling 백업 처리
- ▶ 계층적인 5가지의 로그 메시지 레벨을 사용하며 레벨 별 제어 가능
 - ▶ ① ERROR: 일반 에러가 일어 났을 때 사용
 - ▶ ② WARN: 에러는 아니지만 주의할 필요가 있을 때 사용
 - ▶ ③ INFO: 일반 정보를 나타낼 때 사용
 - ▶ ④ DEBUG: 일반 정보를 상세히 나타낼 때 사용
 - ▶ ⑤ TRACE: 경로추적을 위해 사용

AOP

▶ 관점 지향 프로그래밍(Asspect Oriented Programming)

▶ 실행 시점을 지정할 수 있는 애노테이션

- ▶ **@Before (이전):** 어드바이스 타겟 메소드가 호출되기 전에 어드바이스 기능을 수행
- ▶ **@After (이후):** 타겟 메소드의 결과에 관계없이(즉 성공, 예외 관계없이) 타겟 메소드가 완료 되면 어드바이스 기능을 수행
- ▶ **@AfterReturning (정상적 반환 이후):** 타겟 메소드가 성공적으로 결과값을 반환 후에 어드바이스 기능을 수행
- ▶ **@AfterThrowing (예외 발생 이후):** 타겟 메소드가 수행 중 예외를 던지게 되면 어드바이스 기능을 수행
- ▶ **@Around (메소드 실행 전후):** 어드바이스가 타겟 메소드를 감싸서 타겟 메소드 호출전과 후에 어드바이스 기능을 수행

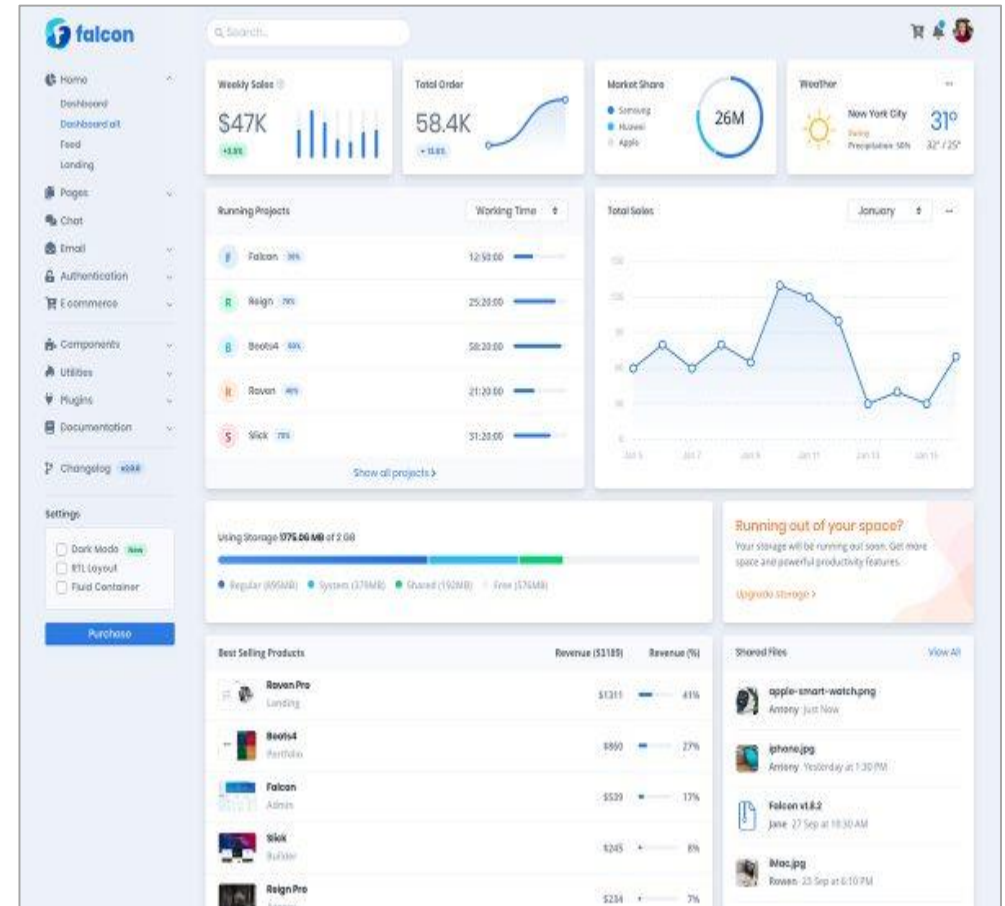


```
@Component
@Aspect
public class LoggerAspect {
    @Pointcut("execution(* com.qhedge.online..controller.*Controller.*(..)) || execution(* com.qhedge.online..service.*Impl.*(..)) || execution(* com.qhedge.online..mapper.*Mapper.*(..))")
    public void getOnline(){}
    @Around("getOnline")
    public Object logPrint(ProceedingJoinPoint joinPoint) throws Throwable {
        String type = "";
        String name = joinPoint.getSignature().getDeclaringTypeName();
        if (name.indexOf("Controller") > -1) { type = ">> Controller : "; }
        } else if (name.indexOf("Service") > -1) { type = ">> ServiceImpl : "; }
        } else if (name.indexOf("Mapper") > -1) { type = ">> Mapper : "; }
        }

        log.debug(type + name + "." + joinPoint.getSignature().getName() + "()");
        return joinPoint.proceed();
    }
}
```

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



Bootstrap Table

- ▶ <https://examples.bootstrap-table.com/>
- ▶ 페이징 처리 및 검색 기능 제공
- ▶ 컬럼의 출력 포맷 및 다양한 속성 지원
- ▶ Ajax(URL), JSON 파일 지원
- ▶ 인쇄 및 파일 저장 기능 제공

田 주식 종목

Search   

| 종목코드 | 종목명 | 상장주식수 | 상장일 |
|--------|----------|-------------|------------|
| 000020 | 동화약품 | 27,931,470 | 1976-03-24 |
| 000040 | KR모터스 | 189,444,075 | 1976-05-25 |
| 000050 | 경방 | 27,415,270 | 1956-03-03 |
| 000060 | 메리츠화재 | 113,680,000 | 1956-07-02 |
| 000070 | 삼양홀딩스 | 8,564,271 | 1968-12-27 |
| 000075 | 삼양홀딩스우 | 304,058 | 1992-02-21 |
| 000080 | 하이트진로 | 70,133,611 | 2009-10-19 |
| 000087 | 하이트진로2우8 | 1,130,138 | 2011-09-26 |
| 000100 | 유한양행 | 12,777,115 | 1962-11-01 |
| 000105 | 유한양행우 | 236,188 | 1995-02-21 |

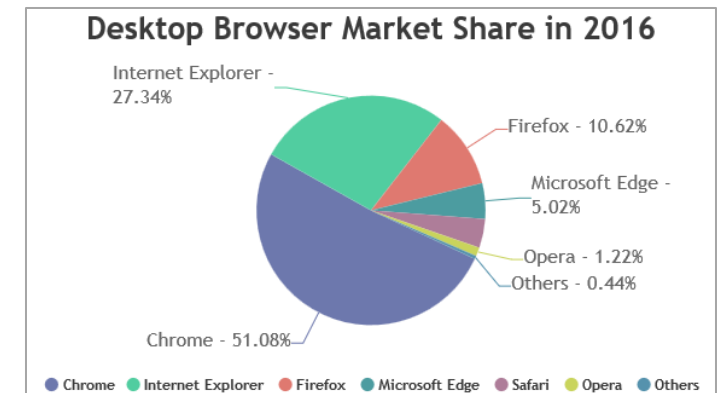
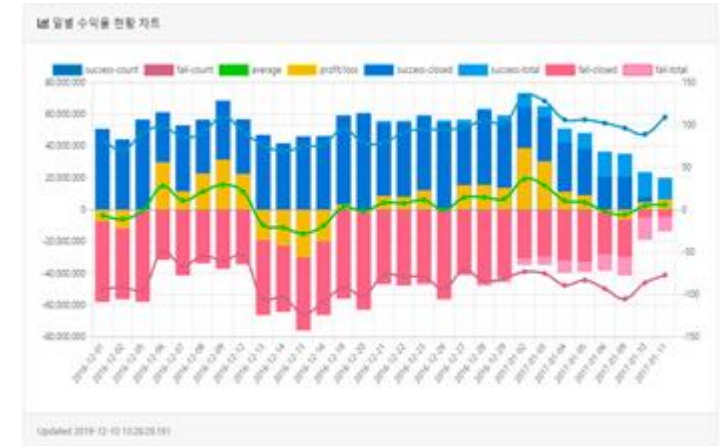
Showing 1 to 10 of 8595 rows 10 rows per page

< 1 2 3 4 5 ... 860 >

Updated 2019-12-10 14:18:40.621

Chart

- ▶ HTML5 기반 JavaScript 형태로 라이브러리 및 샘플 제공
- ▶ 무료/유료 차트 제공 사이트
 - ▶ <http://www.chartjs.org/> 일반적인 간단한 차트그릴 때 유용
 - ▶ <https://developers.google.com/chart/> 구글차트-방대한 기능
 - ▶ <http://private.tistory.com/66> 구글차트 예제
 - ▶ <http://jui.io/?lang=ko> 제니퍼소프트 ui라이브러리
 - ▶ <http://ui.toast.com/tui-chart/> 차트를 포함한 다양한 컴포넌트가 있음
 - ▶ <http://www.jqplot.com/examples/> 순수 jquery chart
 - ▶ <http://dygraphs.com/gallery/#g/stock> 대량의 데이터를 다룰 때 유용
 - ▶ <http://www.flotcharts.org/flot/examples/> Javascript charting library for jQuery
 - ▶ <https://omnipotent.net/jquery.sparkline/#s-about> 작은 차트 그릴 때 유용
 - ▶ <http://tympanus.net/Tutorials/Animated3DBarChart/> 3차원 입체 bar차트
 - ▶ <http://www.amcharts.com/> amchart 로고 있으면 무료
 - ▶ <http://www.highcharts.com/demo/> 막강차트,개인에게는 무료





-END-

