Spring Boot

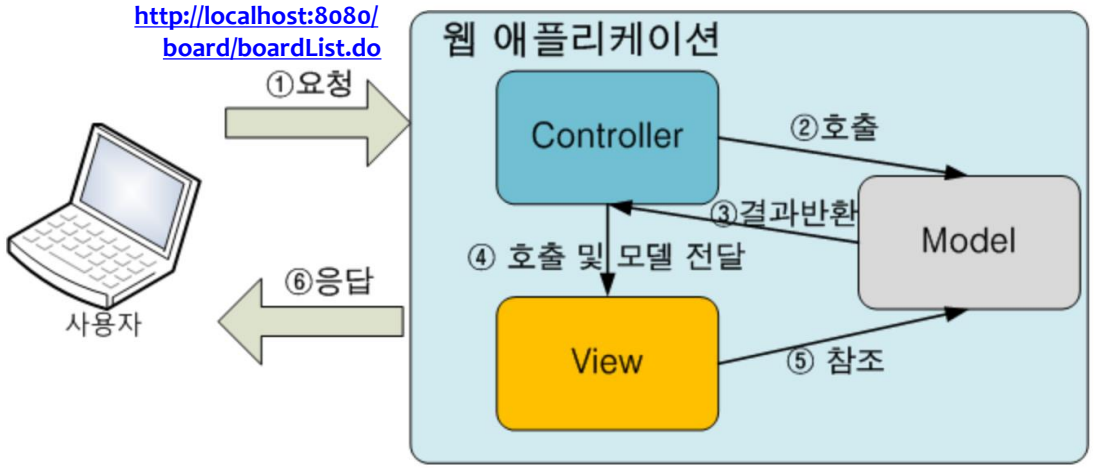
소프트웨어융합공학과
웹서버 프로그래밍
허우행

스프링 프레임워크란?

프로젝트 이름	설명
스프링 프레임워크	현대 자바 기반의 애플리케이션을 개발하는 데 기반이 되는 프레임워크, 스프링의 핵심은 애플리케이션 기반을 제공 함으로써 개발자들은 애플리케이션의 비즈니스 로직의 개발에만 집중할 수 있게 하는데 있다.
스프링 부트	스프링 프레임워크를 기반으로 바로 실행가능한 애플리케이션을 쉽게 만들도록 도와 준다. 대부분의 복잡한 스프링 관련 설정을 자동으로 처리하고 개발자는 최소한의 설정만 진행하면 된다.
스프링 데이터	스프링 애플리케이션에서 다양한 데이터 베이스, JPA 등의 데이터 접근 기술을 쉽게 사용할 수 있도록 도와준다. 스프링 데이터는 하나의 상위 프로젝트로 세부적으로는 데이터베이스 종류에 따라서 수많은 하위 프로젝트가 존재한다.
스프링 시큐리티	자바 애플리케이션에 인증(Authentication) 및 권한(Authorization)에 특화된 프레임워크 이다.
스프링 소셜	페이스북, 트위터, 링크드인과 같은 소셜 서비스 API와 쉽게 연동할 수 있게 도와준다.
※ 이외에도 스프링 클라우드, 스프링 배치, 스프링 모바일 등 다양한 프로젝트가 존재한다.	

Spring Boot 웹(MVC)

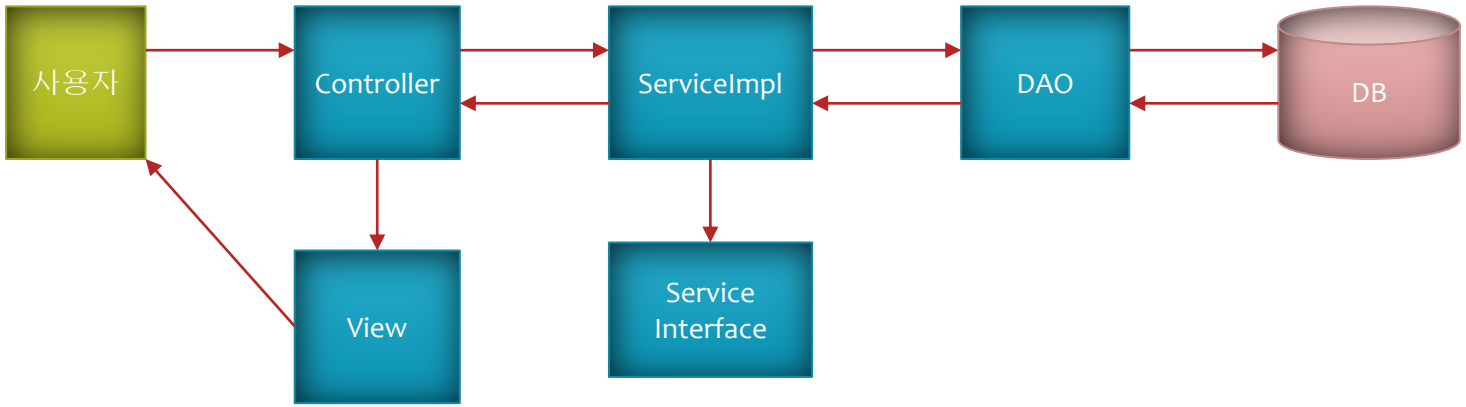
- MVC 패턴을 사용하면 사용자 인터페이스와 비즈니스 로직을 분리하여 개발할 수 있다.
- 서로 영향을 최소화하여 개발 및 변경이 쉬운 애플리케이션을 만들 수 있다.
- MVC 모델
  - Model : 어플리케이션의 정보, 즉 데이터(DB)
  - View : 사용자에게 보여질 화면(html)
  - Controller : 모델과 뷰의 중계역할(html과 데이터 결합)




```
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 웹(MVC)

- MVC 구조
  - View : 사용자가 보는 화면 또는 결과를 의미합니다.
  - Controller : 사용자가 웹브라우저를 통해서 어떠한 요청을 하면 그 요청을 처리할 컨트롤러를 호출하게 됩니다. 컨트롤러는 사용자의 요청을 처리하기 위한 비즈니스 로직을 호출하고 그 결과값을 사용자에게 전달해 주는 역할을 합니다.
  - Service : 사용자의 요청을 처리하기 위한 비즈니스 로직이 수행됩니다. 일반적으로 서비스 영역은 서비스 인터페이스와 이 인터페이스의 구현체로 나뉩니다.
  - DAO : Data Access Object 의 약자로 데이터베이스에 접속해서 비즈니스 로직 실행에 필요한 쿼리를 호출합니다.
  - DB : 데이터베이스를 의미합니다. 데이터베이스에는 애플리케이션에서 발생한 모든 정보가 저장되어 있습니다.



게시판 개발 프로젝트

- 스프링 부트는 웹 애플리케이션의 기능에 따라서 스프링 부트 스타터의 의존성 그룹을 만들고 각각의 그룹에 맞는 의존성이 자동으로 포함되도록 하고 있다.

- 프로젝트 생성 후 build.gradle 파일을 열어 누락된 부분이 없는지 꼼꼼하게 점검한다.

```
dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-aop'          ← aop 는 프로젝트 생성시 선택 불가하며, 수동으로 추가해야 함
    implementation 'org.springframework.boot:spring-boot-starter-jdbc'
    implementation 'org.springframework.boot:spring-boot-starter-thymeleaf'
    implementation 'org.springframework.boot:spring-boot-starter-web'
    implementation 'org.mybatis.spring.boot:mybatis-spring-boot-starter:2.0.0'
    compileOnly 'org.projectlombok:lombok'
    runtimeOnly 'org.springframework.boot:spring-boot-devtools'
    runtimeOnly 'mysql:mysql-connector-java'
    annotationProcessor 'org.springframework.boot:spring-boot-configuration-processor'
    annotationProcessor 'org.projectlombok:lombok'
    providedRuntime 'org.springframework.boot:spring-boot-starter-tomcat'
    testImplementation 'org.springframework.boot:spring-boot-starter-test'
}
```

- 의존성 추가 후에는 Gradle > Refresh Gradle Project 진행하여 필요한 라이브러리를 로컬로 다운로드 한다.

게시판 개발 프로젝트

- src/main/resources/application.properties 에 데이터 소스 설정하기

```
spring.datasource.hikari.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.hikari.jdbc-url=
jdbc:mysql://localhost:3306/board?useUnicode=true&characterEncoding=utf-8&serverTimezone=UTC
spring.datasource.hikari.username=아이디   ← 데이터베이스 설치 시 생성한 ID
spring.datasource.hikari.password=비밀번호   ← 실제 운영 시스템에서는 암호화 처리
spring.datasource.hikari.connection-test-query=SELECT 1   ← 데이터베이스 정상 유무 확인용
```

- 프로퍼티는 프로젝트 전반적인 속성을 관리하는데 사용
- 히카리 CP란?
  - 스프링 부트 2.0.0 M2 부터 기본적으로 사용되는 컨넥션 풀이 히카리CP로 변경되었다.
  - 컨넥션 풀이란 애플리케이션과 데이터베이스를 연결할 때 효과적으로 관리하기 위한 라이브러리 입니다.
  - 데이터베이스로의 추가 요청이 필요할 때 연결을 재사용할 수 있도록 관리되는 데이터베이스 연결 캐시이다
  - 연결 풀을 사용하면 데이터베이스의 명령 실행의 성능을 강화할 수 있다.

게시판 개발 프로젝트

- src/main/java/board/configuration/DatabaseConfiguration.java 클래스 만들기

```
@Configuration
@PropertySource("classpath:/application.properties") ← application.properties 를 읽어오기 위해 지정
Public class DatabaseConfiguration {
      @Bean
      @ConfigurationProperties(prefix="spring.datasource.hikari")  ← 데이터베이스 관련 정보를 사용하도록 지정
      public HikariConfig hikariConfig() {
       return new HikariConfig();
      }

      @Bean
      public DataSource dataSource() throws Exception{  ← 설정 파일을 이용하여 데이터베이스 연결 데이터 소스를 생성
            DataSource dataSource = new HikariDataSource(hikariConfig());
            System.out.println(dataSource.toString());
            return dataSource;
      }
}
```

- 실행결과 로그를 확인하여 정상 동작 상태를 확인한다.

게시판 개발 프로젝트

- src/main/java/board/configuration/DatabaseConfiguration.java 에 마이바티스 설정 추가

```
@Configuration
@PropertySource("classpath:/application.properties")
public class DatabaseConfiguration {
	@Autowired
	private ApplicationContext applicationContext;

	…


	@Bean
	public SqlSessionFactory sqlSessionFactory(DataSource dataSource) throws Exception{
	  SqlSessionFactoryBean sqlSessionFactoryBean = new SqlSessionFactoryBean();
	  sqlSessionFactoryBean.setDataSource(dataSource);
	  sqlSessionFactoryBean.setMapperLocations(applicationContext.getResources("classpath:/mapper/**/sql-*.xml"));  ← 쿼리파일 지정

	  return sqlSessionFactoryBean.getObject();
	}

	@Bean
	public SqlSessionTemplate sqlSessionTemplate(SqlSessionFactory sqlSessionFactory){
	  return new SqlSessionTemplate(sqlSessionFactory);
	}
}
```

- src/main/resources 폴더 밑에 mapper 폴더 생성하고 이곳에 쿼리 파일 보관용

게시판 개발 프로젝트

- src/test/java/BoardApplicationTests.java ← 실행은 Run As → Junit으로 테스트

```
@SpringBootTest
public class BoardApplicationTests {

	@Autowired
	private SqlSessionTemplate sqlSession;

	@Test
	public void contextLoads() {
	}

	@Test
	public void testSqlSession() throws Exception{
	  System.out.println(sqlSession.toString());
	}
}
```

- Console 창에서 에러가 뜨는지 확인하고, SqlSessionTemplate 클래스 출력 확인