JDK 설치

오라클 사이트에서 JDK를 다운받아서 설치합니다.
설치한 후에는 환경 변수에서 JAVA_HOME 을 등록하고 JDK가 설치된 경로를 지정해줍니다.
환경변수의 PATH에는 %JAVA_HOME%bin 으로 저장합니다.
설정이 완료된 뒤에는 명령프롬프트(cmd)에서 javac 명령어를 입력해서 컴파일러가 정상 실행되는지 확인합니다.

이클립스(STS, 전자정부프레임워크) 설정

Lombok 라이브러리의 안정성 때문에 이클립스를 JRE환경이 아닌 JDK환경에서 구동되도록 설정을 변경합니다.
eclipse.ini(혹은 sts.ini) 파일의 상단에 아래와 같은 내용을 추가합니다.

```
-vm
JDK설치경로\bin\javaw.exe
```

이클립스 한글 인코딩(UTF-8) 확인

> *메뉴> Window > Preferences > General > Workspace*
> *메뉴> Window > Preferences > Web > HTML, CSS, JSP 등 확인.*

프로젝트 생성

이클립스에서 spring legacy 프로젝트를 선택하고 spring mvc 프로젝트를 생성합니다.

프로젝트 JDK 버전 변경

여기서는 JDK1.8 과 tomcat9 을 기준으로 작성합니다.

> *Project > Properties > Project Facets 에서 설치된 JDK버전으로 변경합니다.*
> *Project > Properties > Java Compiler 에서 설치된 JDK버전으로 변경합니다.*

Spring MVC 프로젝트 구조 설명

- src/main/java/ : 자바 소스 경로
- src/main/resources/ : 실행시 자동 참고되는 경로(주로 설정파일, log4j.xml 등등)
- src/test/java/ : 테스트 자바 코드 경로
- src/test/resources/ : 테스트 관련 설정 파일 경로
- src/webapp/WEB-INF/spring/appServlet/ : sevlet-context.xml 외 spring 설정 파일
- src/webapp/WEB-INF/spring/ : root-context.xml 외 spring 설정 파일
- src/webapp/WEB-INF/views/ : MVC 패턴 중 view 페이지(jsp) 들이 위치 한 경로
- src/webapp/WEB-INF/ : tomcat의 web.xml 파일 위치

Maven 빌드 도구

maven은 프로젝트에 필요한 의존적인 라이브러리들을 자동으로 관리합니다.

STS나 전자정부프레임워크는 maven을 내장하고 있습니다.
pom.xml 파일을 통해서 maven 설정을 변경 할 수 있습니다.

Spring 및 라이브러리 버전 변경

pom.xml의 수정을 통해 spring의 버전을 변경 할 수 있습니다.
Spring legacy프로젝트로 생성하게 되면 spring3가 기본값이지만 여기서는 jdk1.8, spring5 을 기준으로 합니다.

```
<properties>
<java-version>1.8</java-version>
<org.springframework-version>5.0.7.RELEASE</org.springframework-
version>
<org.aspectj-version>1.6.10</org.aspectj-version>
<org.slf4j-version>1.6.6</org.slf4j-version>
</properties>
```

maven-compiler-plugin 의 JDK버전을 수정 합니다.

```
<plugin>
<groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-compiler-plugin</artifactId>
<version>2.5.1</version>
<configuration>
<source>1.8</source>
<target>1.8</target>
<compilerArgument>-Xlint:all</compilerArgument>
<showWarnings>true</showWarnings>
<showDeprecation>true</showDeprecation>
</configuration>
</plugin>
```

테스트를 위한 spring-test, DB연결을 위한 spring-jdbc, spring-tx 라이브러리를 추가합니다.

```
<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-test</artifactId>
<version>${org.springframework-version}</version>
</dependency>
<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-jdbc</artifactId>
<version>${org.springframework-version}</version>
</dependency>
<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-tx</artifactId>
<version>${org.springframework-version}</version>
</dependency>
```

테스트를 위한 junit 라이브러리의 버전을 변경합니다.

```
<!-- Test -->
<dependency>
<groupId>junit</groupId>
<artifactId>junit</artifactId>
```

```
<version>4.12</version>
<scope>test</scope>
</dependency>
```

servlet 버전을 2.5에서 3.1.0 으로 버전을 변경 합니다.
더불어 jstl, json등 필요한 라이브러리도 추가합니다.

```
<!-- Servlet -->
<dependency>
<groupId>javax.servlet</groupId>
<artifactId>javax.servlet-api</artifactId>
<version>3.1.0</version>
<scope>provided</scope>
</dependency>
<!-- JSTL -->
<dependency>
<groupId>javax.servlet</groupId>
<artifactId>jstl</artifactId>
<version>1.2</version>
</dependency>
<!-- JSON -->
<dependency>
<groupId>com.fasterxml.jackson.core</groupId>
<artifactId>jackson-databind</artifactId>
<version>2.9.6</version>
</dependency>
<dependency>
<groupId>com.fasterxml.jackson.dataformat</groupId>
<artifactId>jackson-dataformat-xml</artifactId>
<version>2.9.6</version>
</dependency>
<dependency>
<groupId>com.google.code.gson</groupId>
<artifactId>gson</artifactId>
<version>2.8.2</version>
</dependency>
```

※ 변경된 프로젝트 pom.xml 설정을 적용하려면 프로젝트를 우클릭 후 Maven > Update Project 를 실행합니다.

Lombok 라이브러리 설치

https://projectlombok.org 에서 jar 파일을 다운 받을 수 있습니다.
Lombok은 bean객체로 쓰이는 클래스들을 @Data 어노테이션 등으로 손쉽게 관리해줍니다.
명령프롬프트에서 java -jar lombok.jar 으로 실행합니다.
이클립스 실행파일 경로를 지정해주면 해당 경로에 lombok.jar 가 설치된 것을 확인 할 수 있습니다.
기존 이클립스 바로가기를 삭제 후 바로가기를 다시 생성하십시오.
그리고 pom.xml에 아래와 같이 추가합니다.

```
<!-- Lombok -->
<dependency>
<groupId>org.projectlombok</groupId>
<artifactId>lombok</artifactId>
<version>1.18.0</version>
<scope>provided</scope>
</dependency>
```

lombok.jar 를 이클립스 프로젝트의 build path에 libraries 에 추가하고 (Web) Deployment Assembly 항목에도 해당 jar 파일을 추가합니다. 그래야 WAS에 배포시에도 적용이 됩니다.

Tomcat 설치

http://tomcat.apache.org 에서 다운 받을 수 있습니다.
이클립스 메뉴에서 windows > preferences > server > Runtime Evironments 에서 해당 tomcat이 설치된 폴더를 지정합니다.

Database 연결 세팅

데이터베이스 무료 라이센스가 필요하다면 오픈소스 마리아DB https://mariadb.org/ 을 이용해 보십시오.
오라클의 경우 학습용은 무료지만 상업용 서비스는 유료입니다.
여기서는 오라클 11g Express Edition을 기준으로 합니다.
https://www.oracle.com/downloads/index.html JDBC 드라이버는 현재 버전에 맞는 드라이버를 사용합니다.
(여기서는 ojdbc8.jar 를 사용합니다.)
오라클 jdbc드라이버는 maven에서 지원하지 않으므로 수동으로 JDBC드라이버를 이클립스 프로젝트의 build path에 libraries 에 추가하고 (Web) Deployment Assembly 항목에도 해당 jar 파일을 추가합니다.
그래야 WAS에 배포시에도 적용이 됩니다.
그리고 Database 스키마의 문자셋 UTF-8 등과 프로젝트의 문자셋을 확인하십시오.

커넥션 풀(DataSource) 설정

spring-jdbc 라이브러리를 사용하는 방식도 있지만 여기서는 HikariCP 라이브러리를 사용하겠습니다.
pom.xml 수정

```
<dependency>
<groupId>com.zaxxer</groupId>
<artifactId>HikariCP</artifactId>
<version>2.7.4</version>
</dependency>
```

root-context.xml 수정: DB설정 정보는 자신의 설정에 맞도록 수정합니다.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:context="http://www.springframework.org/schema/context"
xsi:schemaLocation="http://www.springframework.org/schema/beans
https://www.springframework.org/schema/beans/spring-beans.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-4.3.xsd">
<!-- Root Context: defines shared resources visible to all other web components -->
<bean id="hikariConfig" class="com.zaxxer.hikari.HikariConfig">
<property name="driverClassName" value="oracle.jdbc.driver.OracleDriver"></property>
<property name="jdbcUrl" value="jdbc:oracle:thin:@localhost:1521:orcl"></property>
<property name="username" value="scott"></property>
<property name="password" value="tiger"></property>
</bean>
<bean id="dataSource" class="com.zaxxer.hikari.HikariDataSource" destroy-method="close">
<constructor-arg ref="hikariConfig" />
</bean>
<context:component-scan base-package="example.package.com"></context:component-scan
</beans>
```

만약 spring의 dataSource를 사용한다면 아래를 참고해 root-context.xml을 수정 하십시오.

```
<bean id="dataSource"
class="org.springframework.jdbc.datasource.DriverManagerDataSource">
<property name="driverClassName"
value="net.sf.log4jdbc.sql.jdbcapi.DriverSpy"></property>
<property name="url" value="jdbc:oracle:thin@localhost:1521/orcl">
</property>
<property name="username" value="scott"></property>
<property name="password" value="tiger"></property>
</bean>
```

MyBatis 설정

pom.xml에 MyBatis 와 log4jdbc 라이브러리를 추가합니다.

```
<!-- Mybatis -->
<dependency>
<groupId>org.mybatis</groupId>
<artifactId>mybatis</artifactId>
<version>3.4.6</version>
</dependency>
<dependency>
<groupId>org.mybatis</groupId>
<artifactId>mybatis-spring</artifactId>
<version>1.3.2</version>
</dependency>
<!-- log4jdbc -->
<dependency>
<groupId>org.bgee.log4jdbc-log4j2</groupId>
<artifactId>log4jdbc-log4j2-jdbc4</artifactId>
<version>1.16</version>
</dependency>
```

SQLSessionFactory 설정: root-context.xml 을 수정합니다. dataSource bean 다음에 추가하면 됩니다.

```
<!-- Mybatis Session Factory -->
<bean id="sqlSessionFactory"
class="org.mybatis.spring.SqlSessionFactoryBean">
<property name="dataSource" ref="dataSource"></property>
</bean>
```

Mapper 설정: SQL mapper(SQL과 Class를 mapping) 인터페이스들이 저장되어 있는 패키지를 스캔할 수 있도록 root-context.xml 설정이 필요합니다.
먼저 root-context.xml의 아래쪽 Namespaces 탭에 mybatis-spring 항목을 체크합니다.
그리고 root-context.xml에 scan 할 패키지를 아래처럼 지정해줍니다.

```
<mybatis-spring:scan base-package="패키지이름.mapper"/>
```

XML 파일을 SQL Mapper로 사용하려면 XML파일의 위치와 namespace 속성이 중요한데 Mapper인터페이스와 같은 경로로 하거나 src/main/resources/ 에 xml들을 저장할 폴더를 생성하고 xml파일을 작성합니다.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="패키지이름.매퍼인터페이스이름">
```

```
<!--매퍼 인터페이스에 있는 메서드 이름과 동일하게 id를 지정, resultType은 메서드의 리턴타입을 지정 -->
<select id="getTime" resultType="string">
SELECT sysdate FROM dual
</select>
</mapper>
```

src/main/resources/ 경로에 log4jdbc.log4j2.properties 파일을 생성 한 뒤 아래와 같이 입력합니다.

```
log4jdbc.spylogdelegator.name=net.sf.log4jdbc.log.slf4j.Slf4jSpyLogDelegator
```

한글 (UTF-8) 인코딩 필터

web.xml 에 아래와 같이 한글 처리 필터를 추가합니다.

```
<filter>
<filter-name>encoding</filter-name>
<filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
<init-param>
<param-name>encoding</param-name>
<param-value>UTF-8</param-value>
</init-param>
</filter>
<filter-mapping>
<filter-name>encoding</filter-name>
<servlet-name>appServlet</servlet-name>
</filter-mapping>
```