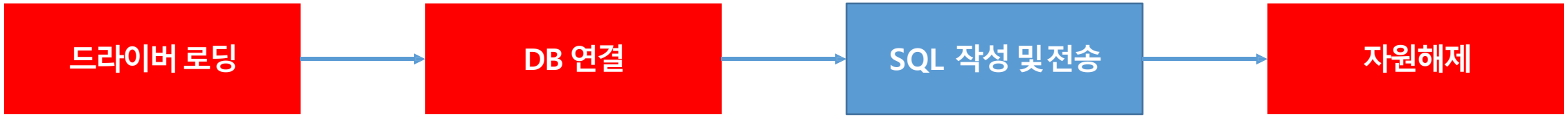
Spring Framework	-스프링 MVC웹서비스 1.JDBC 2.Spring-JDBC(템플릿)

1 :JDBC

JDBC



전통적인 JDBC프로그래밍
1. Connection 객체 생성
2. PrepareStatement 객체생성
3. SQL문 실행
4. ResultSet객체 생성 결과처리

너무 반복되는 작업이 계속되는 단점이 있다

2. Spring-JDBC

Spring JDBC란? (**JdbcTemplate**)

- JDBC의 장점을 유지하면서, 전통적방식의 JDBC단점을 극복하여, 간결한 형태의 API 사용법을 제공하며 기존 방식에서 지원하지 않는 편리한 기능을 제공

- Spring JDBC는 반복적으로 하는 작업을 대신함
  (connection, prepareSatement, resultSet, resultSet의 반복처리, Exception처리)

- Spring JDBC는 SQL에 바인딩할 값을 지정만 해주면 된다.
- Spring JDBC 사용 전 DB커넥션을 가져오는 DataSource가 강제화 된다.

2. Spring-JDBC

커넥션 풀

- 여러 명의 사용자를 동시에 처리하는 웹 어플리케이션

- DB연결을 이용할 때 매번 연결하는 방식이 아닌 미리 연결을 맺고 사용하는 Connection Pool을 이용해 성능을 향상시킴

- 커넥션 풀링은 미리 정해진 개수만큼 DB커넥션을 풀에 준비해두고, 어플리케이션이 요청할 때마다 Pool에서 꺼내서 할당하며, 다시 돌려 받아서 Pool에 넣는 기법

속도면에서 빠르며, 최근 유행하는 HikariCP 라이브러리 를 사용

DataSource

- DB에 이용되는 URL, id, pw, DriverClass 를 미리 정의해 놓고 사용 하는 객체

- Spring-Jdbc에서 기본으로 제공
- 여러 커넥션풀 라이브러리에서 기본으로 제공

2. Spring-JDBC

Select 구문

query메서드

```
query(sql, new Object[] {}, new RowMapper<Type>() {

        익명클래스
})
```

sql - ?를 사용하는 preparedStatement사용

new Object[] {값, 값}
- sql물음표 값을 세팅할 값을 저장

new RowMapper<Type>()
- 조회 결과를 ResultSet으로 읽어 Type으로 반환
- 익명클래스 사용
- mapRow()메서드를 오버라이딩 해서 사용

insert update, delete 구문

update메서드

```
update(sql, new Object[] {값, 값, 값});
```

sql - ?를 사용하는 preparedStatement사용

new Object[] {값, 값}
- sql물음표 값을 세팅할 값을 저장(바인딩)

2. Spring-JDBC

최종: 커넥션풀은 Hikari 라이브러리 를 이용

스프링 JDBC를 하기위한 필요 라이브러리
메이븐에 추가(pom.xml)

1. Mysql Connector(커넥터)
2. Spring-JDBC
3. 히카리CP (커넥션풀)
4. Spring-test(테스트)

DB연결을 테스트하기 위해 사용: 선택사항

라이브러리 객체 생성
스프링 설정파일에 bean 추가

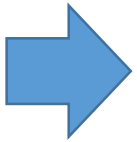
- 히카리 빈 설정
- DataSource 빈 설정
- JdbcTemplate 빈 설정

2. Spring-JDBC

pom.xml 설정

Mysql Connector 라이브러리

```
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.15</version>
</dependency>
```



Spring-JDBC 라이브러리

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

히카리 라이브러리

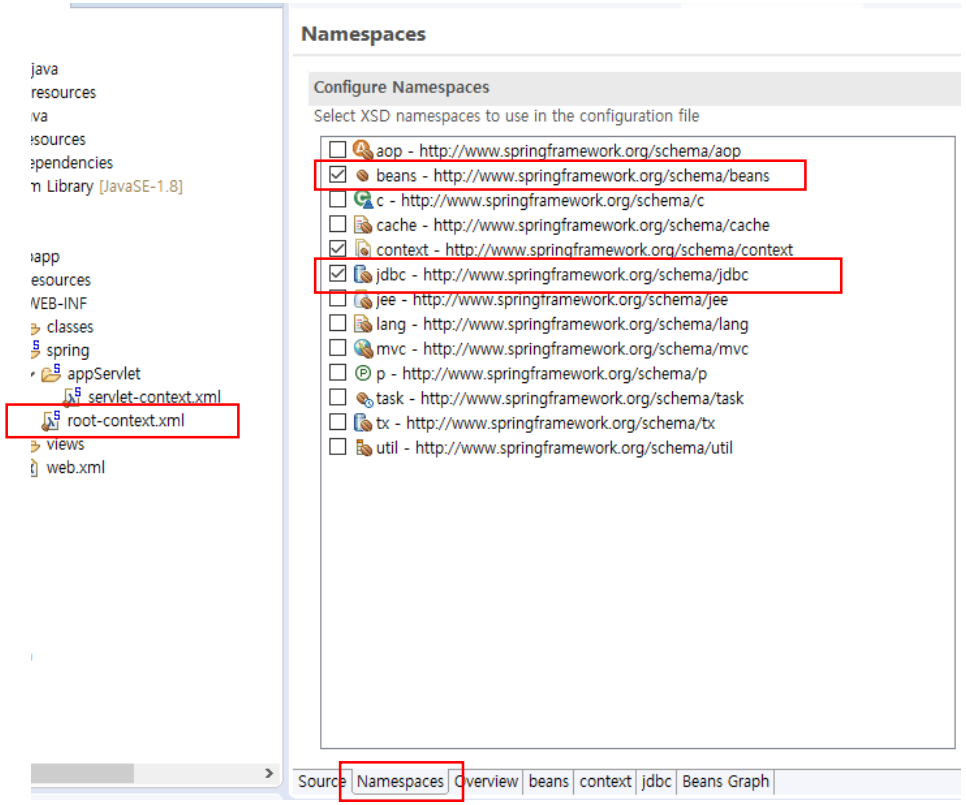
```
<dependency>
        <groupId>com.zaxxer</groupId>
        <artifactId>HikariCP</artifactId>
        <version>2.7.4</version>
</dependency>
```

Spring-test(테스트) 라이브러리

```
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-test</artifactId>
    <version>5.0.7.RELEASE</version>
    <scope>test</scope>
</dependency>
```

2. Spring-JDBC

DataSource와 HikariCP 빈으로 등록 하기 위한 namespace추가

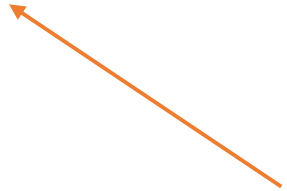


2. DataSource설정

DataSource와 HikariCP 빈으로 등록

```
<bean id="hikariConfig" class="com.zaxxer.hikari.HikariConfig">
        <property name="driverClassName" value="com.mysql.cj.jdbc.Driver"></property>
        <property name="jdbcUrl"
                        value="jdbc:mysql://localhost:3306/spring?serverTimezone=Asia/Seoul"></property>
        <property name="username" value="spring"></property>
        <property name="password" value="spring"></property>
</bean>


<bean id="dataSource" class="com.zaxxer.hikari.HikariDataSource">
        <constructor-arg ref="hikariConfig"></constructor-arg>
</bean>
```



코드 해석

-> hikariConfig이름으로 컨테이너에 객체 생성
-> 세터 주입으로 name에 value를 저장

코드 해석

-> Class속성에 정의된 클래스를 dataSource 이름으로 컨테이너에 객체생성
-> 생성자 주입으로 위에 선언한 hikariConfig를 주입
-> ref는 참조속성 입니다.

2. Spring-JDBC

JdbcTemplate을 사용하기 위한
Root-context.xml 설정 추가

```
<!-- jdbcTemplate이름으로 객체생성하며 의존주입 -->
<bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
        <property name="dataSource" ref="dataSource"></property>
</bean>
```

코드 해석

-> Class속성에 정의된 클래스를 jdbcTemplate이름으로 컨테이너에 객체생성
-> dataSource 메서드에 앞서 생성한 dataSource를 주입
-> ref는 참조속성 입니다.

3. 추가: HikariCP 주요 옵션들

주요 속성명	기능
autoCommit	자동commit설정 (기본:true)
connectionTimeout	pool에서 커넥션을 얻어오기전까지 기다리는 최대 시간 (기본:30초)
maximumPoolSize	pool에 유지시킬 수 있는 최대 커넥션 수 (기본:10개)
maxLifetime	커넥션 풀에서 살아있을 수 있는 커넥션의 최대 수명시간. (기본 30분)

참고 라이브러리

https://github.com/brettwooldridge/HikariCP#configuration-knobs-baby