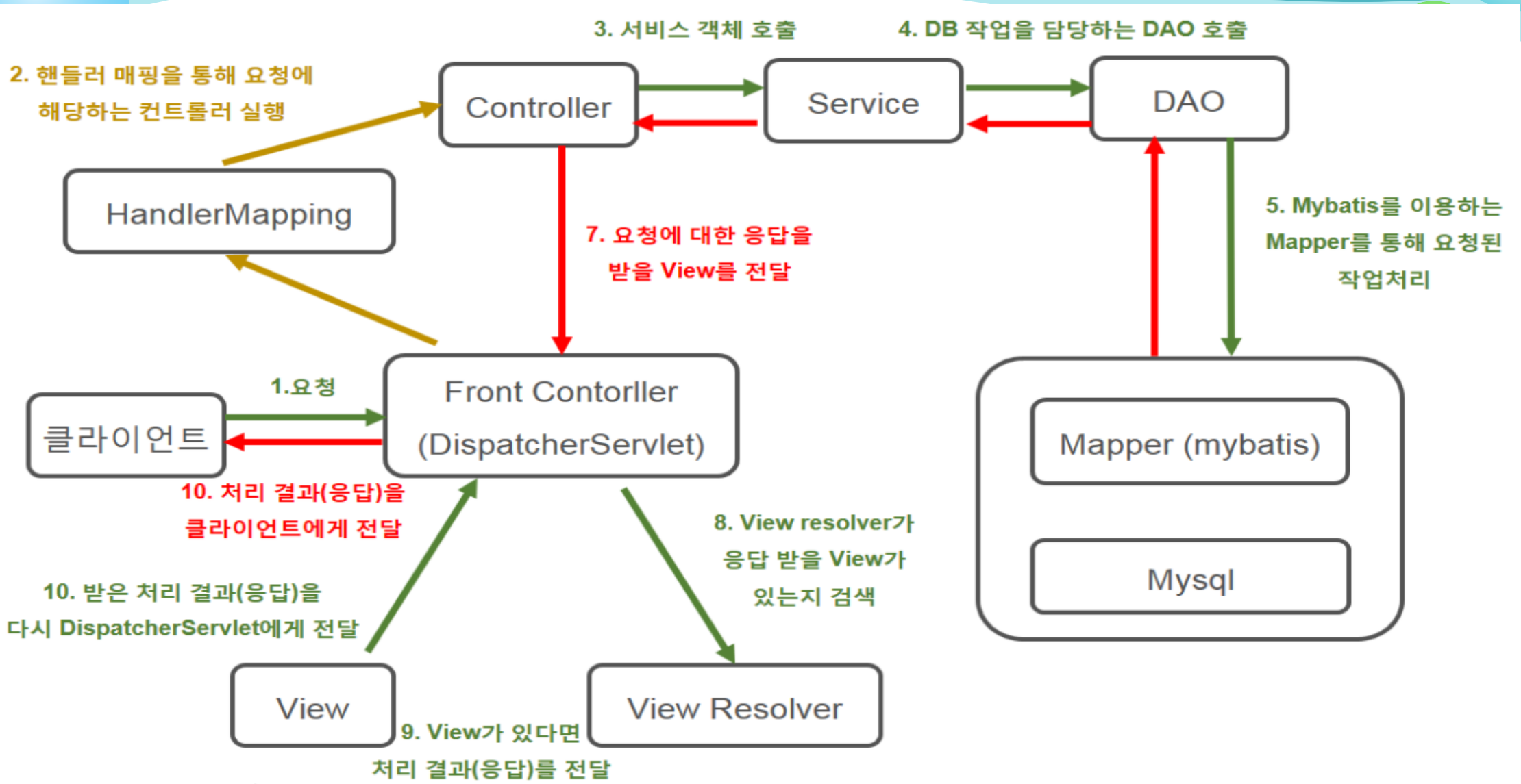
MyBatis 연동

안 화 수

Spring MVC 흐름도



DAO class를 이용한 MyBatis – Spring 연동

myBatis2

- myBatis2 project를 import 해보자

- 환경 설정 파일
  1. pom.xml (maven의 환경 설정 파일)
  2. web.xml (project의 환경 설정 파일)
  3. servlet-context.xml (spring의 환경 설정 파일)
  4. root-context.xml (spring의 환경 설정 파일)
  5. configuration.xml (mybatis의 환경 설정 파일)
  6. Mapper파일 (Dept.xml, Emp.xml)

pom.xml (1/3)

- pom.xml 에 의존 라이브러리 추가 : ojdbc8.jar

```
<dependencies>
        <!-- Oracle -->
        <dependency>
                <groupId>com.oracle.database.jdbc</groupId>
                <artifactId>ojdbc8</artifactId>
                <version>21.5.0.0</version>
        </dependency>
</dependencies>
```

pom.xml (2/3)

- pom.xml 에 의존 라이브러리 추가 : Connection Pool기능 - HikariCP

```
<dependencies>
        <!-- Hikari CP-->
        <dependency>
                <groupId>com.zaxxer</groupId>
                <artifactId>HikariCP</artifactId>
                <version>2.7.4</version>
        </dependency>
</dependencies>
```

pom.xml (3/3)

- pom.xml 에 의존 라이브러리 추가 : mybatis, mybatis-spring

```
<dependencies>
		<!-- mybatis -->
		<dependency>
			<groupId>org.mybatis</groupId>
			<artifactId>mybatis</artifactId>
			<version>3.5.6</version>
		</dependency>
		<dependency>
			<groupId>org.mybatis</groupId>
			<artifactId>mybatis-spring</artifactId>
			<version>1.3.2</version>
		</dependency>
</dependencies>
```

web.xml (1/3)

- **web.xml에 DispatcherServlet 설정**
- DispatcherServlet 을 등록하고 servlet mapping을 설정한다.
- Spring의 환경 설정 파일을 등록한다.

```
<servlet>
        <servlet-name>appServlet</servlet-name>
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
        <init-param>
                <param-name>contextConfigLocation</param-name>
                <param-value>/WEB-INF/spring/appServlet/servlet-context.xml</param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
        <servlet-name>appServlet</servlet-name>
        <url-pattern>*.do</url-pattern>
</servlet-mapping>
```

web.xml (2/3)

- **web.xml에 Spring의 환경 설정 파일 등록**
- Spring의 환경 설정 파일을 등록한다.
- DataBase 접속과 같이 공통적인 내용을 root-context.xml 파일에 등록한다.

```
<!-- The definition of the Root Spring Container shared by all Servlets and Filters -->
<context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>/WEB-INF/spring/root-context.xml</param-value>
</context-param>
<!-- Creates the Spring Container shared by all Servlets and Filters -->
<listener>
        <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
</listener>
```

web.xml (3/3)

- **web.xml에 한글 인코딩을 처리하기 위한 필터 등록**
  - 한글값이 POST 방식으로 전송될때 한글이 깨지지 않도록 인코딩을 처리하기 위한 필터를 등록한다.

```
<filter>
        <filter-name>CharacterEncodingFilter</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>
        <init-param>
                <param-name>forceEncoding</param-name>
                <param-value>true</param-value>
        </init-param>
</filter>
<filter-mapping>
        <filter-name>CharacterEncodingFilter</filter-name>
        <url-pattern>/*</url-pattern>
</filter-mapping>
```

Spring의 환경 설정 파일(servlet-context.xml)

❖ Spring의 환경 설정 파일 : servlet-context.xml

- Spring의 환경 설정 파일 ViewResolver 를 등록한다.
  1. view 파일을 저장할 최상위 디렉토리(prefix) : /WEB-INF/views/
  2. view 파일의 확장자(suffix) : .jsp

- Spring의 환경 설정 파일에 base-package 를 등록한다.
  1. base-package 하위 클래스를 스캔한다는 의미를 가진다.
  2. base-package 하위 클래스에 @Controller, @Service, @Repository 어노테이션이 붙어있는 클래스는 @Autowired 어노테이션을 이용해서 필요한 빈 객체를 setter 메소드 없이 자동으로 주입을 받는다.

```
<beans:bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <beans:property name="prefix" value="/WEB-INF/views/" />
        <beans:property name="suffix" value=".jsp" />
</beans:bean>

<context:component-scan base-package="myBatis2" />
```

Spring의 환경 설정 파일(root-context.xml)

- Spring의 환경 설정 파일 : root-context.xml (1/2)

```
<beans>
	<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:xe"></property>
		<property name="username" value="scott"></property>
		<property name="password" value="tiger"></property>
	</bean>

	<!-- HikariCP configuration -->
	<bean id="dataSource" class="com.zaxxer.hikari.HikariDataSource" destroy-method="close">
		<constructor-arg  ref="hikariConfig" />
	</bean>
</beans>
```

Spring의 환경 설정 파일(root-context.xml)

- Spring의 환경 설정 파일 : root-context.xml (2/2)

```
<beans>
	<!-- 스프링으로 oracle db 연결 -->
	<bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
		<property name="dataSource"  ref="dataSource"></property>
		<property name="configLocation"  value="classpath:configuration.xml" />
		<property name="mapperLocations"  value="classpath:sql/*.xml" />
	</bean>

	<bean id="session" class="org.mybatis.spring.SqlSessionTemplate">
		<constructor-arg index="0“  ref="sqlSessionFactory" />
	</bean>
</beans>
```

Mybatis의 환경설정파일(configuration.xml)

- Mybatis의 환경설정파일 : configuration.xml

```
<configuration>

        <!– DTO 클래스의 alias를 설정 -->
        <typeAliases>
                <typeAlias alias="dept" type="myBatis2.model.Dept" />
                <typeAlias alias="emp"  type="myBatis2.model.Emp" />
        </typeAliases>

        <!– 생략한다. (spring의 환경설정 파일(root-context.xml)에서 설정한다.)
        <mappers>
                <mapper resource="sql/Dept.xml" />
                <mapper resource="sql/Emp.xml" />
        </mappers>  -->

</configuration>
```

Dao class

- **Dao class : DeptDaoImpl.java**

mapper 파일(Dept.xml) 파일의 id를 불러서 SQL문을 실행한다.

```
package myBatis2.dao;

import java.util.List;

@Repository
public class DeptDaoImpl implements DeptDao {
    @Autowired
    private SqlSessionTemplate st;

    public List<Dept> list() {
        return st.selectList("deptns.list");
    }

    public int insert(Dept dept) {
        return st.insert("deptns.insert", dept);
    }

    public Dept select(int deptno) {
        return st.selectOne("deptns.select", deptno);
    }

    public int update(Dept dept) {
        return st.update("deptns.update", dept);
    }

    public int delete(int deptno) {
        return st.delete("deptns.delete", deptno);
    }
}
```

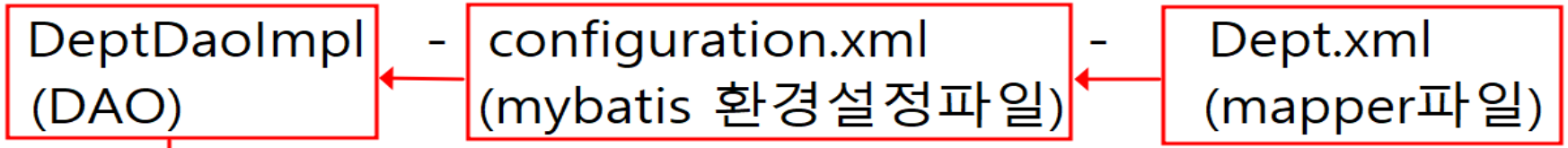
mapper파일

mapper : Dept.xml

mapper 파일마다 서로 다른 namespace를 설정하고, 태그들은 id값을 설정한다.

```
<mapper namespace="deptns">

    <!-- Use type aliases to avoid typing the full classname every time. -->
    <resultMap id="deptResult"    type="dept">
        <result property="deptno" column="deptno" />
        <result property="dname"  column="dname" />
        <result property="loc"    column="loc" />
    </resultMap>

    <select id="list" resultMap="deptResult">
        select * from dept order by deptno
    </select>

    <select id="select" parameterType="int" resultType="dept">
        select * from dept where deptno=#{deptno}
    </select>

    <update id="update" parameterType="dept">
        update dept set dname=#{dname},loc=#{loc} where deptno=#{deptno}
    </update>

    <delete id="delete" parameterType="int">
        delete from dept where deptno=#{deptno}
    </delete>

    <insert id="insert" parameterType="dept">
        insert into dept values(#{deptno},#{dname},#{loc})
    </insert>

</mapper>
```

Dao – MyBatis 연동



SqlSession (SQL문을 실행시키는 메소드를 제공하는 인터페이스)

SqlSessionTemplate (SqlSession을 상속받는 구현클래스)

SQL문 - 메소드

---

insert - int insert()

update - int update()

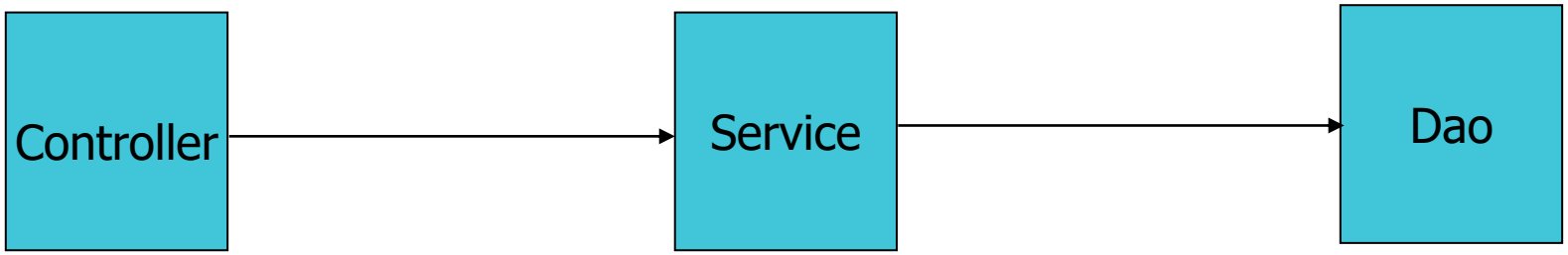
delete - int delete()

select - Object selectOne() : 검색 결과가 1개인 경우에 사용

List selectList() : 검색 결과가 여러개인 경우에 사용

Controller – Service - Dao

Controller – Service - Dao



@Controller
DeptController

@Autowired
private DeptService service

@Service
DeptService

@Autowired
private DeptDao dao;

@Repository
DeptDao

@Autowired
private SqlSession session

mapper interface를 이용한 MyBatis – Spring 연동

myBatis3 project를 import 해보자

환경 설정 파일

1. pom.xml (maven의 환경 설정 파일)
2. web.xml (project의 환경 설정 파일)
3. servlet-context.xml (spring의 환경 설정 파일)
4. root-context.xml (mapper interface package 추가)
5. configuration.xml (mybatis의 환경 설정 파일)
6. Mapper파일 (Dept.xml, Emp.xml) – namespace에 mapper interface의 path추가
7. DAO 클래스 대신에 mapper interface로 mapper 파일을 실행한다.
   - mapper interface의 method명과 mapper파일의 id값을 같은 이름으로 설정한다.

Spring의 환경 설정 파일(root-context.xml)

- **Spring의 환경 설정 파일 : root-context.xml**
  mapper interface의 package를 설정한다.

```
<beans>

        <!-- mapper interface의 package 설정 -->
        <mybatis-spring:scan base-package="myBatis3.mapper"/>

</beans>
```

mapper interface

- **mapper interface : DeptDao.java**

  mapper 파일(Dept.xml) 파일의 id명과 같은 이름의 method로 작성한다.

```java
package myBatis3.mapper;

import java.util.List;

//@Mapper
public interface DeptDao {
    List<Dept> list();

    int insert(Dept dept);

    Dept select(int deptno);

    int update(Dept dept);

    int delete(int deptno);
}
```

mapper파일

- **mapper : Dept.xml**

namespace는 mapper interface의 path를 설정하고, id값은 mapper interface의 method명으로 설정한다.

```
<mapper namespace="myBatis3.mapper.DeptDao">

    <!-- Use type aliases to avoid typing the full classname every time. -->
    <resultMap id="deptResult"    type="dept">
        <result property="deptno" column="deptno" />
        <result property="dname"  column="dname" />
        <result property="loc"    column="loc" />
    </resultMap>

    <select id="list" resultMap="deptResult">
        select * from dept order by deptno
    </select>

    <select id="select" parameterType="int" resultType="dept">
        select * from dept where deptno=#{deptno}
    </select>

    <update id="update" parameterType="dept">
        update dept set dname=#{dname},loc=#{loc} where deptno=#{deptno}
    </update>

    <delete id="delete" parameterType="int">
        delete from dept where deptno=#{deptno}
    </delete>

    <insert id="insert" parameterType="dept">
        insert into dept values(#{deptno},#{dname},#{loc})
    </insert>

</mapper>
```

mapper interface

- **mapper interface : EmpDao.java**

mapper 파일(Emp.xml) 파일의 id명과 같은 이름의 method로 작성한다.

```
package myBatis3.mapper;

import java.util.List;

//@Mapper
public interface EmpDao {
    List<Emp> list(int deptno);

    List<Emp> empList();

    Emp select(int empno);

    int insert(Emp emp);

    int delete(int empno);

    int update(Emp emp);

    List<Emp> empAllList();
}
```

mapper파일

- **mapper : Emp.xml**

namespace는 mapper interface의 path를 설정하고, id값은 mapper interface의 method명으로 설정한다.

```
<mapper namespace="myBatis3.mapper.EmpDao">

    <!-- Use type aliases to avoid typing the full classname every time. -->
    <resultMap id="empResult"       type="emp">
        <result property="empno"    column="empno" />
        <result property="ename"    column="ename" />
        <result property="job"      column="job" />
        <result property="mgr"      column="mgr" />
        <result property="hiredate" column="hiredate" />
        <result property="sal"      column="sal" />
        <result property="comm"     column="comm" />
        <result property="deptno"   column="deptno" />
        <result property="dname"    column="dname" />
        <result property="loc"      column="loc" />
    </resultMap>

    <select id="empList" resultMap="empResult">
        select * from emp order by empno
    </select>

    <select id="list" parameterType="int" resultMap="empResult">
        select * from emp where deptno=#{deptno} order by empno
    </select>

    <select id="select" parameterType="int" resultType="emp">
        select * from emp where empno=#{empno}
    </select>
```