spring jdbc

1. pom.xml 에 spring jdbc 찾아서 추가

```
<!-- https://mvnrepository.com/artifact/org.springframework/spring-jdbc -->
<dependency>
	<groupId>org.springframework</groupId>
	<artifactId>spring-jdbc</artifactId>
	<version>4.1.4.RELEASE</version>
</dependency>
```

2. servlet-context.xml 추가

```
<beans:bean name="dataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
	<beans:property name="driverClassName" value="com.mysql.jdbc.Driver" />
	<beans:property name="url" value="jdbc:mysql://localhost:3306/spring" />
	<beans:property name="username" value="root" />
	<beans:property name="password" value="1111" />
</beans:bean>
<beans:bean name="template" class="org.springframework.jdbc.core.JdbcTemplate">
	<beans:property name="dataSource" ref="dataSource" />
</beans:bean>
```

3. util 패키지 Constant 클래스 추가

```
public class Constant {
	public static JdbcTemplate template;
}
```

4. Controller에서 @Autowired 작업

```
private JdbcTemplate template
@Autowired
public void setTemplate(JdbcTemplate template) {
	this.template = template;
	Constant.template = this.template;
}
```

5. ContactDao 작업

```
JdbcTemplate template;
public BDao() {
	this.template = Constant.template;
}
```

```java
// ContactListCommand
public ArrayList<ContactDTO> list() {
	String sql = "select * from contact order by idx desc";
	return (ArrayList<ContactDTO>) template.query(sql, new BeanPropertyRowMapper<ContactDTO>(ContactDTO.class));
}

// ContactWriteCommand
public void write(final String name, final String tel, final String addr, final String email, final String relative) {
	template.update(new PreparedStatementCreator() {
		@Override
		public PreparedStatement createPreparedStatement(Connection con) throws SQLException {
			String sql = "insert into contact(name, tel, addr, email, relative) values(?, ?, ?, ?, ?)"
			PreparedStatement ps = con.prepareStatement(sql);
			ps.setString(1, name);
			ps.setString(2, tel);
			ps.setString(3, addr);
			ps.setString(4, email);
			ps.setString(5, relative);
			return ps;
		}
	});
}

// ContactViewCommand
public ContactDTO view(int idx) {
	String sql = "select * from contact where idx=" + String.valueOf(idx);
	return template.queryForObject(sql, new BeanPropertyRowMapper<ContactDTO>(ContactDTO.class));
}

// ContactModifyCommand
public void modify(final int idx, final String name, final String tel, final String addr, final String email, final String relative) {
	String sql = "update contact set name=?, tel=?, addr=?, email=?, relative=? where idx=?"
	template.update(sql, new PreparedStatementSetter() {
		@Override
		public void setValues(PreparedStatement ps) throws SQLException {
			ps.setString(1, name);
			ps.setString(2, tel);
			ps.setString(3, addr);
			ps.setString(4, email);
			ps.setString(5, relative);
			ps.setInt(6, idx);
		}
	});
}

// ContactDeleteCommand
public void delete(final int idx) {
	String sql = "delete from contact where idx=?";
	template.update(sql, new PreparedStatementSetter() {
		@Override
		public void setValues(PreparedStatement ps) throws SQLException {
			ps.setInt(1, idx);
		}
	});
}
```