*079. Spring IoC 컨테이너 도입

	요청	/service/*	서블릿↓		2. getBean( )	"Bean 컨테이너, DI 컨테이너"		생성	
client	→	<<front Controller>> **DispatcherServlet**			→	**Spring IoC 컨테이너**		→	**컨트롤러**
	←				↘ 4. call		3. 객체 리턴		**서비스**
- refresh	응답(redirect)		↑			**컨트롤러**	↓		**DAO**
- redirect		**JSP**		- 경과 페이지 생성					
- jsk 실행결과				- 오류 내용 출력					
- 오류 내용									

* "Bean" (커피콩)   *IoC (Inversion of Control) = "역제어", "제어의 역전"
"Object"   1. 일반적인 경우 : 객체가 필요하면 생성해서 쓴다.
"instance"   └ 역제어 : 필요한 객체를 외부에서 만들어 주입해준다.   ← 객체를 교체하기 쉽다. "Dependency Injection" 외존 객체 주입 (DI)
        2. 일반적인 경우 : 어떤 작업을 수행하기 위해 메서드를 호출
        └ 역제어 : 특정 상태에 놓일 때 등록된 메서가 자동 호출된다. = "리스너, 필터, 서블릿 등"   IoC
        DI
        listener
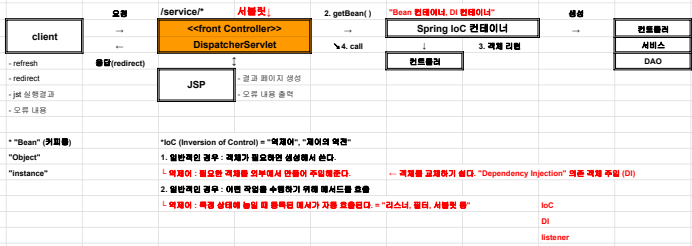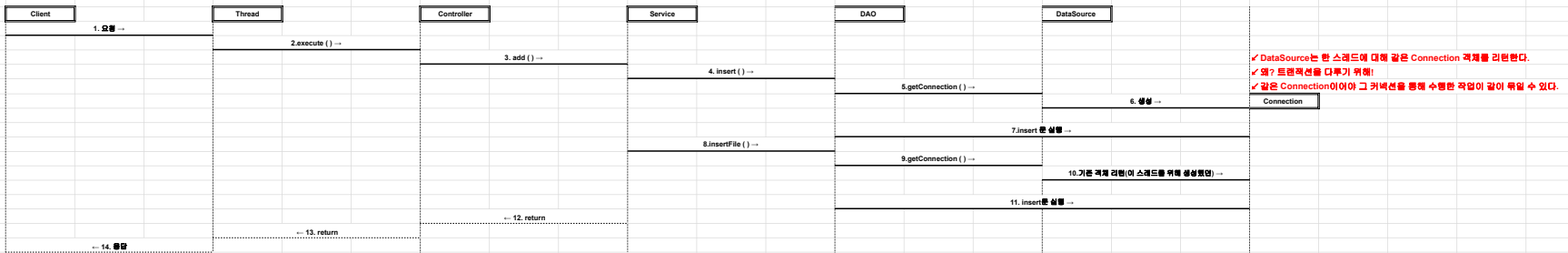
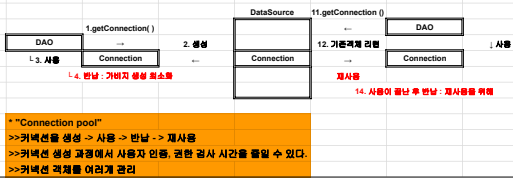https://search.maven.org/artifact/org.springframework/spring-context/5.3.23/jar
implementation 'org.springframework:spring-context:5.3.23'
spring-context 라이브러리 검색
의존 라이브러리 정보를 추가한다.

@ComponentScan
//내가 패키지이름 알려줄테니깐 패키지 및 하위 패키지를 다 뒤져서, 내가 알아볼 수 있는 애노테이션이 있는 클래스를 다 찾아내
//그래서 그 클래스 객체를 다 생성해.

---

*DataSource 의 역할 1 : 같은 스레드 같은 커넥션

Client		Thread		Controller		Service		DAO		DataSource	
	1. 요청 →										
		2.execute ( ) →									
				3. add ( ) →							
						4. insert ( ) →					
								5.getConnection ( ) →			
										6. 생성 →	**Connection**
								7.insert 문 실행→			
						8.insertFile ( ) →					
								9.getConnection ( ) →			
										10.기존 객체 리턴(이 스레드를 위해 생성했던) →	
								11. insert문 실행 →			
				← 12. return							
		← 13. return									
	← 14. 응답										

✔ DataSource는 한 스레드에 대해 같은 Connection 객체를 리턴한다.
✔ 왜? 트랜잭션을 다루기 위해!
✔ 같은 Connection이어야 그 커넥션을 통해 수행한 작업이 같이 묶일 수 있다.

---

*DataSource의 역할2 : 커넥션 재사용

		DataSource	11.getConnection ()			
	1.getConnection( )		←	**DAO**		
DAO	→		12. 기존객체 리턴	→	↓ 사용	
└ 3. 사용	**Connection**	**Connection**	→	**Connection**		
			재사용			
└ 4. 반납 : 가비지 생성 최소화			14. 사용이 끝난 후 반납 : 재사용을 위해			

* "Connection pool"
>>커넥션을 생성 -> 사용 -> 반납 -> 재사용
>>커넥션 생성 과정에서 사용자 인증, 권한 검사 시간을 줄일 수 있다.
>>커넥션 객체를 여러개 관리

---

*DataSource의 커넥션 풀 구동 과정

		DataSource			
Thread1	1. 요청 →		5. 요청	**Thread3**	
Connection1	2. 생성 ←	**Connection1**	6. 생성 →	**Connection3**	
└ 반납					
Thread2	3. 요청 →	**Connection2**	7. 요청 ←	**Thread4**	
Connection2	4. 생성 ←		8. 생성	**Connection2**	
└ 반납					

			*커넥션 요청		
Thread5	9. 요청 →		1. 없으면, 새로 만들어 준다.	DVD 대여와 유사!	
			2. 있으면, 기존 객체를 준다.		
Connection1	10. 생성 ←		└ 생성에 소모되는 시간을 줄일 수 있다. (사용자 인증, 권한 검사)		
			└ 가비지 생성을 줄일 수 있다. (메모리를 효율적으로 관리)		

```xml
<bean id="dataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
    <property name="driverClassName" value="${jdbc.driverClassName}"/>
    <property name="url" value="${jdbc.url}"/>
    <property name="username" value="${jdbc.username}"/>
    <property name="password" value="${jdbc.password}"/>
</bean>

<context:property-placeholder location="jdbc.properties"/>
```

*Spring Framework와 라이브러리 파일

Spring Freamwork	**1. Spring - context = Spring IoC 컨테이너와 관련된 라이브러리**
	2. Spring - jdbc = Spring JDBC 관련 라이브러리

implementation group: 'org.springframework', name: 'spring-jdbc', version: '5.3.23'

*Spring Ioc 컨테이너 준비과정
new AnnotationConfigApplicationContext(AppConfig.class)
1.AppConfig(자바 config) 객체 생성

★해보기)055. 클래스 정보를 알아내서 애노테이션 정보를 꺼내서 생성사 객체를 통해서 인스턴스를 만드는것. = miniwebserver
string objname = clazz.~~
constructor<?>~
objmap.~~~

*Spring webMVC
implementation 'org.springframework:spring-webmvc:5.3.23'
Context는 제거한다. MVC가 의존하고 있음.

*Spring WebMvc의 DispathcerServerlet