토비의 스프링 부트

이해와 원리

2022.1.20

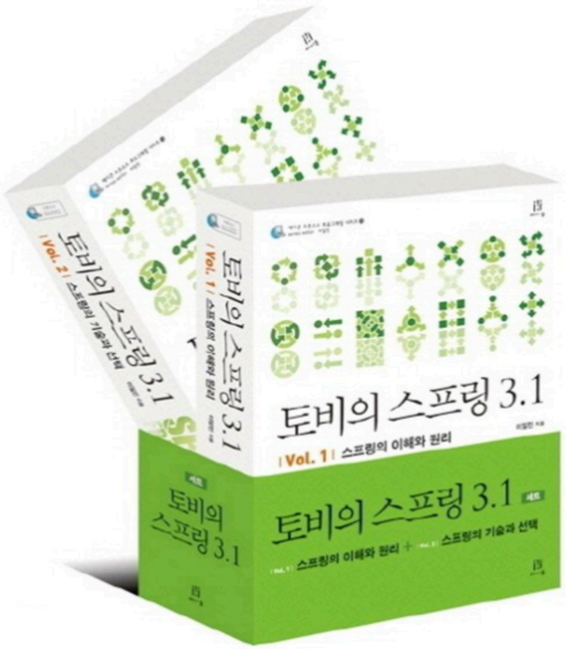
강의 소개

토비의 스프링 부트 - 이해와 원리

스프링 부트의 핵심 기능을 코드로 직접 구현하면서 스프링 부트의 동작 원리와 스프링 부트에 적용된 스프링 프레임워크의 활용법을 익히게 되는 강의

강사 소개

Toby 이일민



토비의 스프링 3.0, 3.1 저자

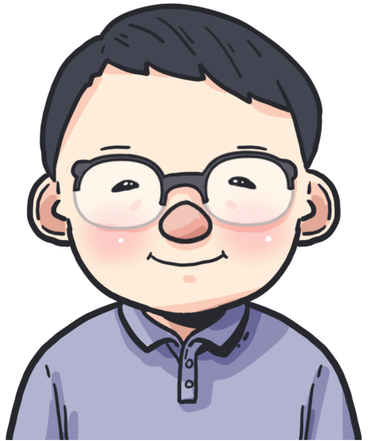
한국스프링사용자모임(KSUG) 설립자

스프링 기반의 시스템 개발 20년 경력

유튜브 기술 채널 (www.youtube.com/@tobyspring)

디스코드 서버 (https://discord.gg/x4eT5HYk6X)

이메일: tobyilee@gmail.com



기획동기와 학습방법

토비의 스프링 부트 - 이해와 원리는 어떻게 만들어졌나

스프링 부트를 만나고 나서 갑자기 스프링 초보자가 된 듯 당황한 토비!
스프링 부트가 어떻게 동작하는지, 어떤 원리로 만들어졌는지를 이해하기위해 노력했고,
스프링 부트가 스프링을 어떻게 이용해서 동작하는지를 코드를 통해서 설명할 수 있는 방법을 연구해서 강의로 만들었습니다

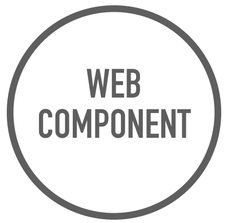
스프링 부트 시작하기

Containerless

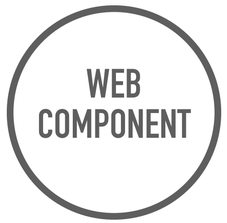
컨테이너리스 웹 애플리케이션 아키텍처

Serverless

Container

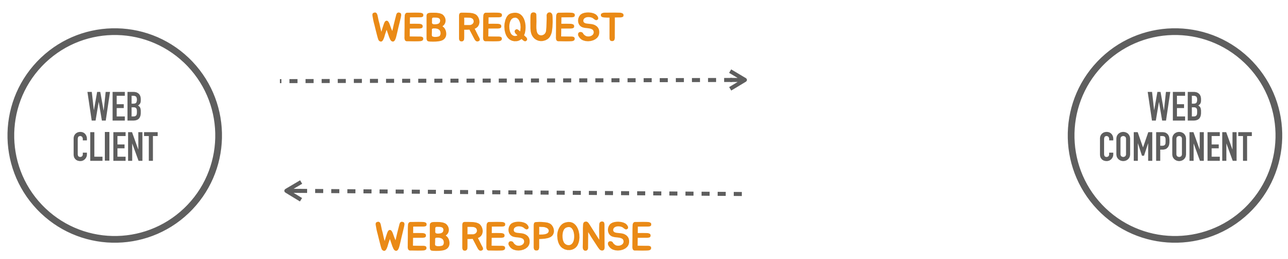

WEB
COMPONENT

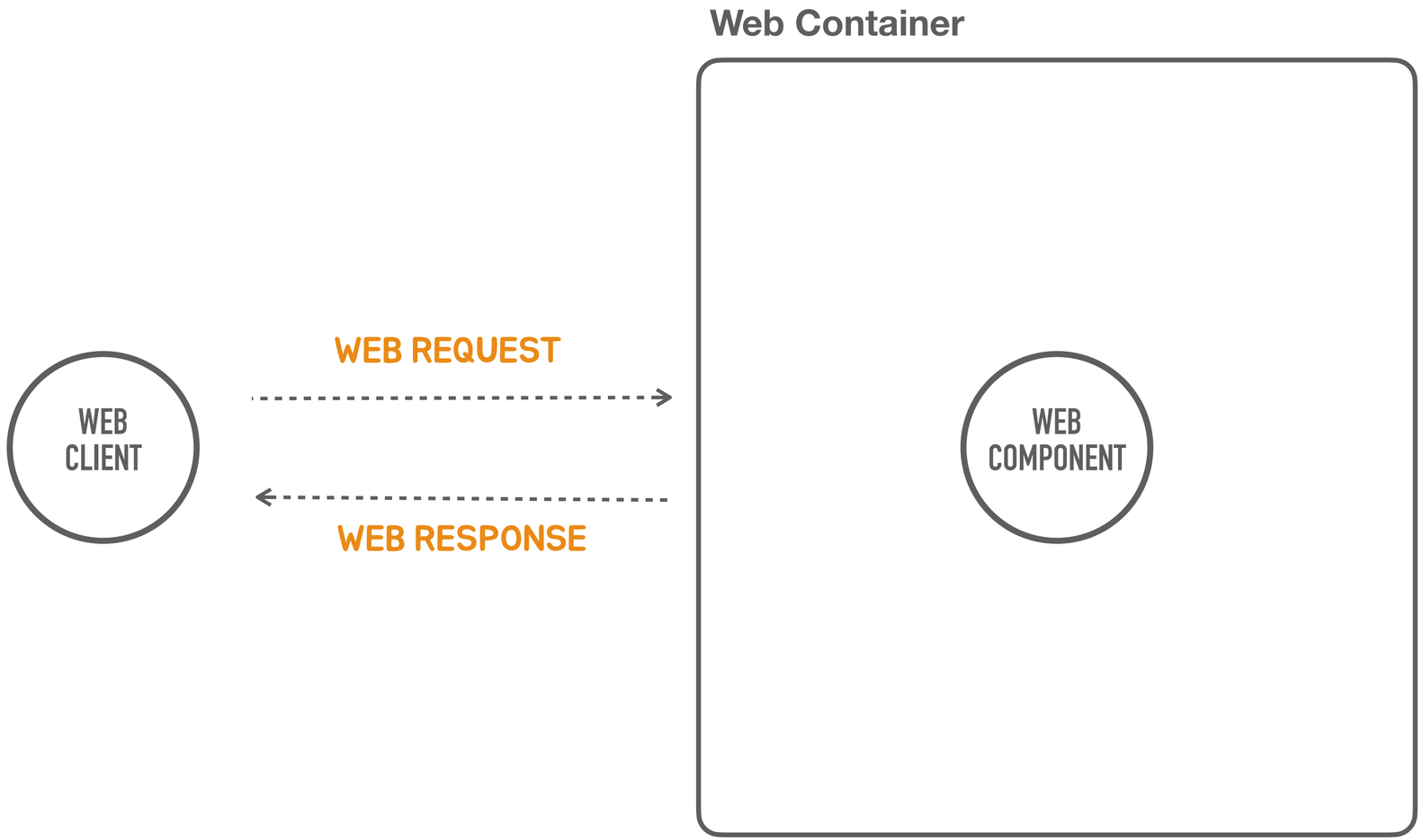

WEB
CLIENT

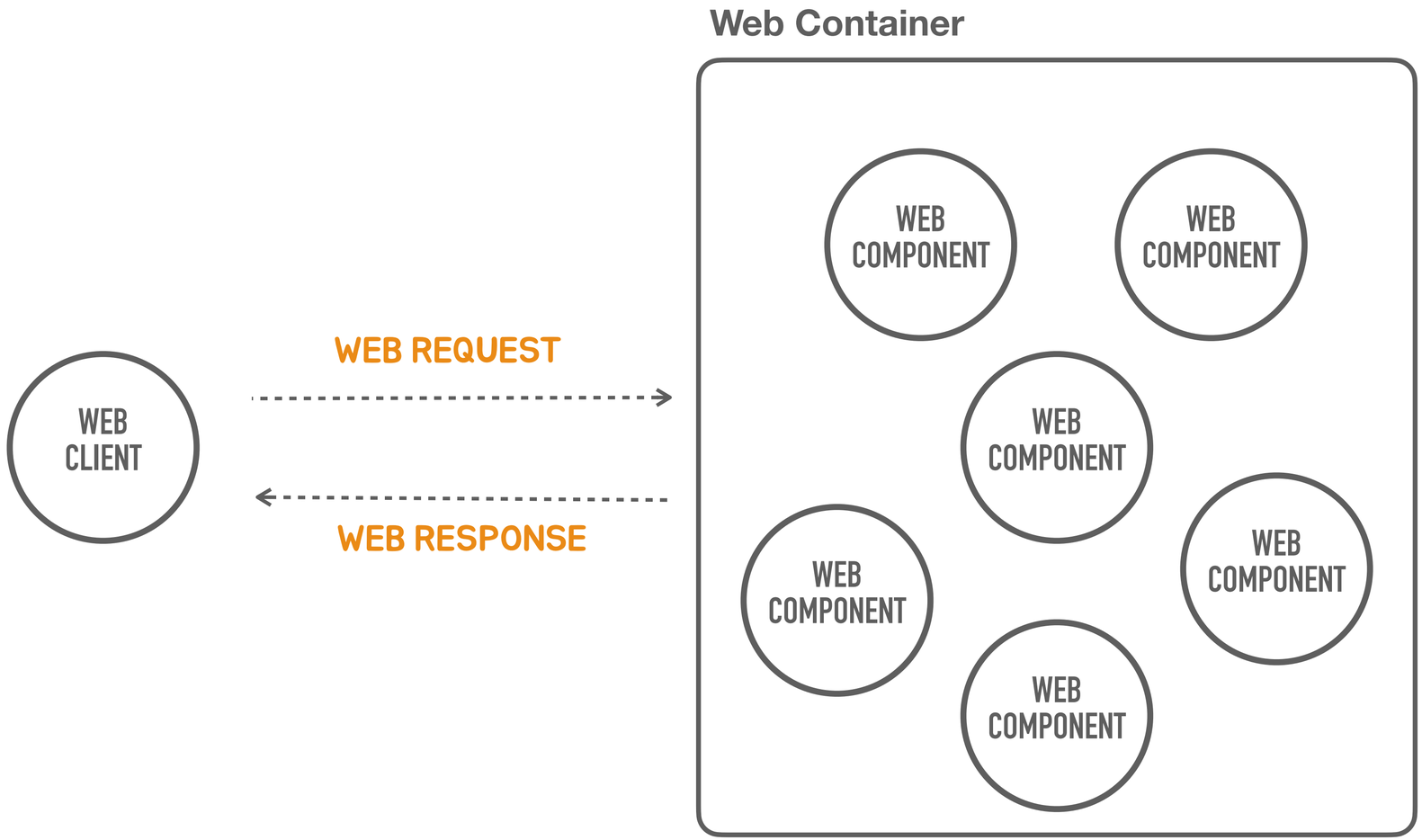

WEB
COMPONENT

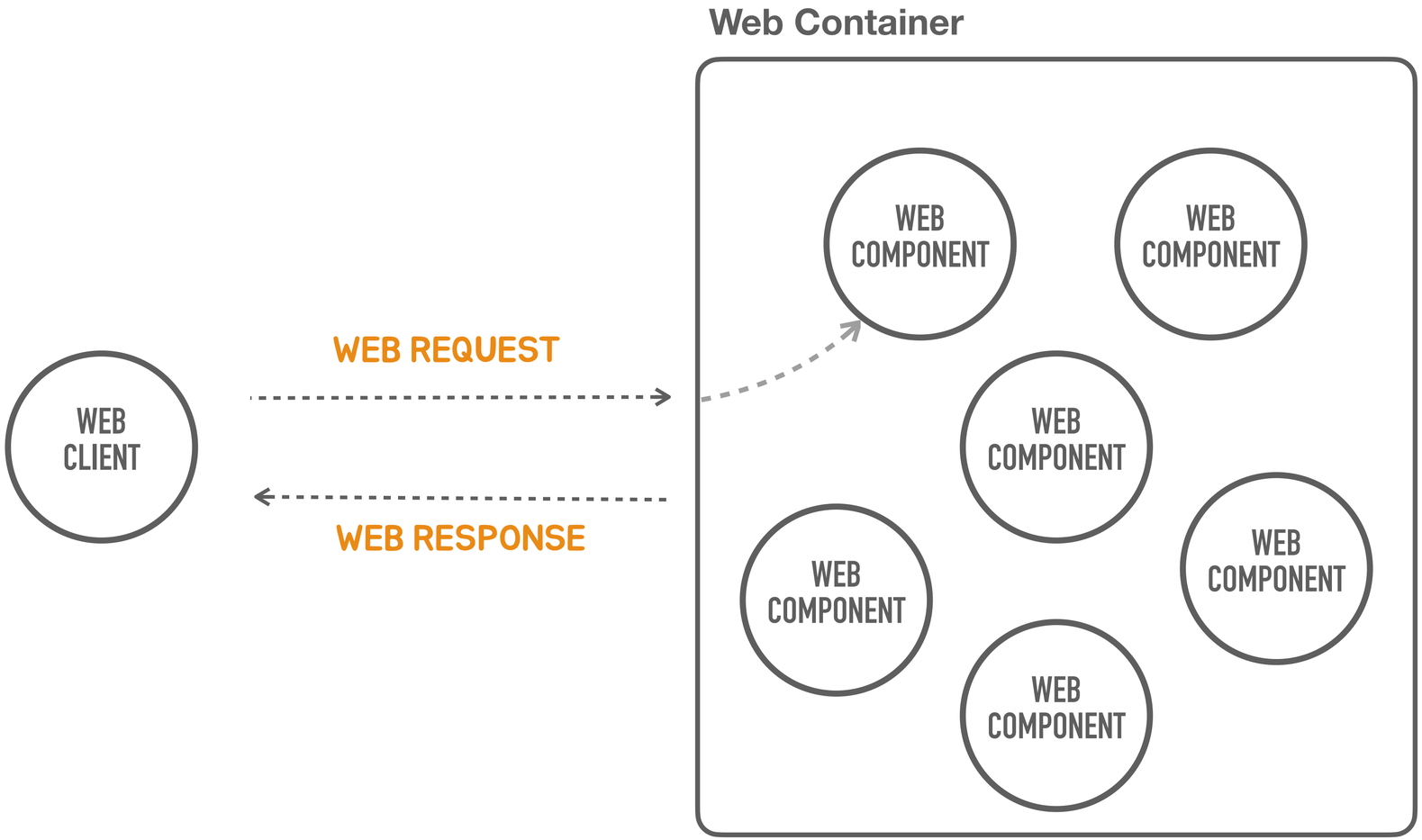

WEB REQUEST
WEB CLIENT
WEB COMPONENT

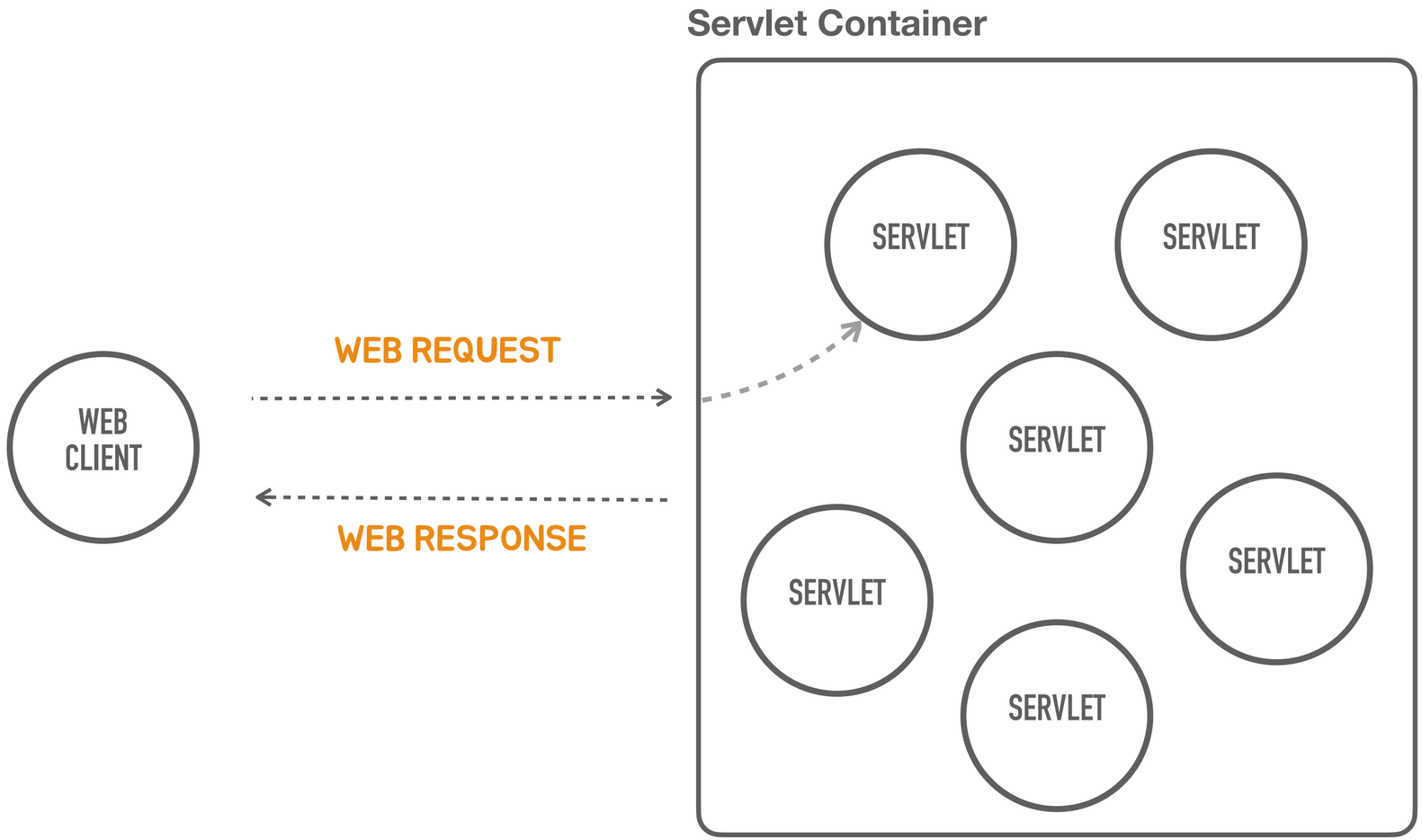

WEB REQUEST
WEB CLIENT
WEB COMPONENT
Dynamic Content

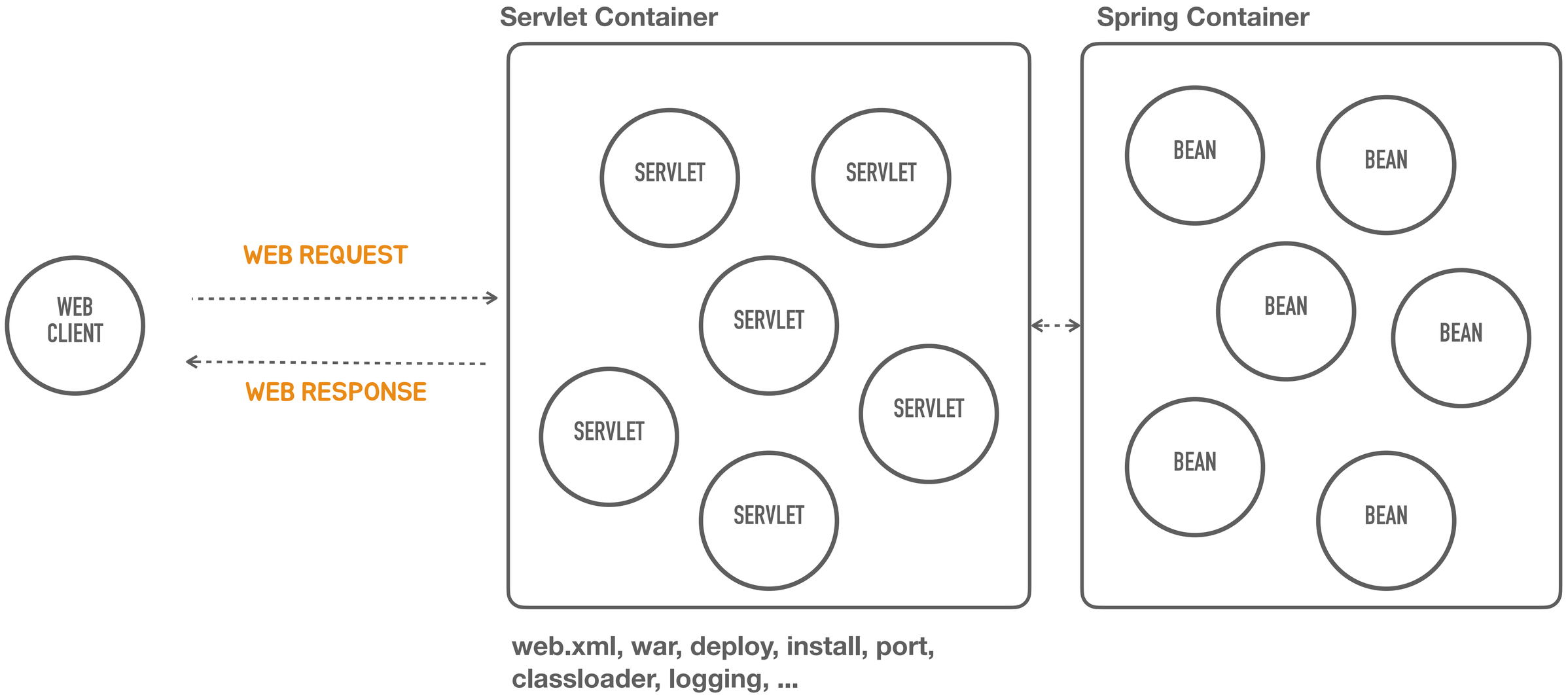

WEB REQUEST
WEB CLIENT
WEB COMPONENT
WEB RESPONSE

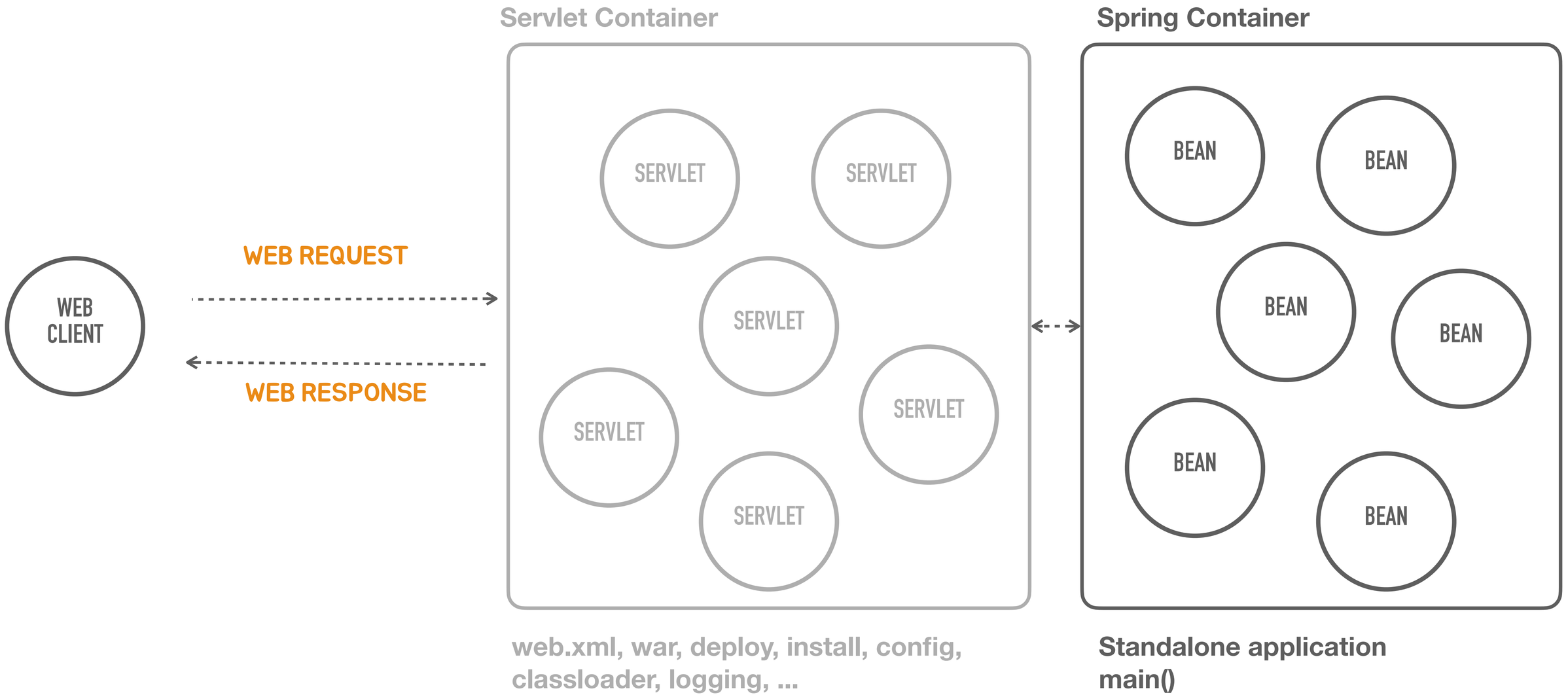

Web Container
WEB REQUEST
WEB CLIENT
WEB COMPONENT
WEB RESPONSE


Web Container
WEB CLIENT
WEB REQUEST
WEB RESPONSE
WEB COMPONENT
WEB COMPONENT
WEB COMPONENT
WEB COMPONENT
WEB COMPONENT
WEB COMPONENT


Web Container
WEB CLIENT
WEB REQUEST
WEB RESPONSE
WEB COMPONENT
WEB COMPONENT
WEB COMPONENT
WEB COMPONENT
WEB COMPONENT
WEB COMPONENT


Servlet Container
WEB CLIENT
WEB REQUEST
WEB RESPONSE
SERVLET
SERVLET
SERVLET
SERVLET
SERVLET
SERVLET


Servlet Container
Spring Container
WEB CLIENT
WEB REQUEST
WEB RESPONSE
SERVLET
SERVLET
SERVLET
SERVLET
SERVLET
SERVLET
BEAN
BEAN
BEAN
BEAN
BEAN
BEAN
web.xml, war, deploy, install, port,
classloader, logging, ...


Servlet Container
Spring Container
WEB CLIENT
WEB REQUEST
WEB RESPONSE
SERVLET
SERVLET
SERVLET
SERVLET
SERVLET
SERVLET
BEAN
BEAN
BEAN
BEAN
BEAN
BEAN
web.xml, war, deploy, install, config, classloader, logging, ...
Standalone application main()

```
<web-app>

    <listener>
        <listener-class>org.springframework.web.context.ContextLoaderListener</listener-
class>
    </listener>

    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>/WEB-INF/root-context.xml</param-value>
    </context-param>

    <servlet>
        <servlet-name>app1</servlet-name>
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
        <init-param>
            <param-name>contextConfigLocation</param-name>
            <param-value>/WEB-INF/web-context.xml</param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
    </servlet>

    <servlet-mapping>
        <servlet-name>app1</servlet-name>
        <url-pattern>/app1/*</url-pattern>
    </servlet-mapping>

</web-app>
```

Opinionated

내가 다 정해줄게 일단 개발만 해

스프링 프레임워크의 설계 철학

- 극단적인 유연함 추구
- 다양한 관점을 수용
- Not opinionated
- 수많은 선택지를 다 포용
- 하지만...

스프링 부트의 설계 철학

- Opionionated - 자기 주장이 강한, 자기 의견을 고집하는, 독선적인
- 일단 정해주는 대로 빠르게 개발하고 고민은 나중에
- 스프링을 잘 활용하는 뛰어난 방법을 제공

사용 기술과 의존 라이브러리 결정

- 업계에서 검증된 스프링 생태계 프로젝트, 표준 자바 기술, 오픈소스 기술의 종류와 의존관계, 사용 버전을 정해줌
- 각 기술을 스프링에 적용하는 방식(DI 구성)과 디폴트 설정값 제공

Spring 3.0 Dependencies - 1 Oct 2009

File Edit View Insert Format Data Tools Extensions Help

A1 groupId

	groupId	artifactId	version	aop	asm	aspects	beans	context	ctx.supp	core	expr	inst	inst.cl	ument.tor	jdbc	jms	orm	oxm	test	tx	web	portlet	servlet
2	org.springframework	aop						X								X	O			X	X		
3	org.springframework	asm		T			X	X		O												T	T
4	org.springframework	beans		X		T		X	X							X	X	X	X	O	X	X	X	X
5	org.springframework	context				T			X							O	X	O	X	O	X	X	X	X
6	org.springframework	context.support				T																		O
7	org.springframework	core		X			X	X	X		X					X	X	X	X	O	X	X	X	X
8	org.springframework	expression						X																
9	org.springframework	instrument						O																
10	org.springframework	jdbc							O									X		O				
11	org.springframework	orm																		O				T
12	org.springframework	oxm															O					O		O
13	org.springframework	test				T																		
14	org.springframework	transaction				O			O							X	O	X		O				T
15	org.springframework	web																O		O		X	X	
16	org.springframework	web.portlet																		O				
17	org.springframework	web.servlet																		O			X	
18	org.beanshell	bsh	2.0.0.b4					O																
19	com.caucho	com.caucho	3.2.1																			O		
20	org.codehaus.woodstox	com.ctc.wstx	3.2.7							T														
21	org.apache.ibatis	com.ibatis	2.3.4.726															O						
22	com.ibm.websphere	com.ibm.websphere.uow	6.0.2.17																		O			
23	com.jamonapi	com.jamonapi	2.4.0	O																				
24	com.lowagie.text	com.lowagie.text	2.0.8																					O
25	com.mchange.c3p0	com.mchange.v2.c3p0	0.9.1.2												O									
26	com.sun.enterprise	com.sun.enterprise.loader	1.0.0					P																
27	com.sun.syndication	com.sun.syndication	1.0.0																					O
28	com.thoughtworks.xstream	com.thoughtworks.xstream	1.3.0																O					
29	com.bea.commonj	commonj	1.1.0						O															
30	edu.emory.mathcs.backport	edu.emory.mathcs.backport	3.0.0					O																
31	org.freemarker	freemarker	2.3.15						O															O
32	javax.activation	javax.activation	1.1.0						P											P				
33	javax.annotation	javax.annotation	1.0.0					O																
34	javax.ejb	javax.ejb	3.0.0					O													O			
35	javax.el	javax.el	1.0.0				P													P		O	P	P
36	javax.faces	javax.faces	1.2.0.08																			O		
37	javax.inject	javax.inject	0.9.0.PFD				P	P																
38	javax.jdo	javax.jdo	2.1.0															O						
39	javax.jms	javax.jms	1.1.0					O								P								
40	javax.mail	javax.mail	1.4.0			T			O															
41	javax.persistence	javax.persistence	1.0.0					O										O		P	T			
42	javax.portlet	javax.portlet	2.0.0																	P		P	P	
43	javax.resource	javax.resource	1.5.0														O				O			
44	javax.servlet	javax.servlet	2.5.0							T								O		P		P	X	P
45	javax.servlet	javax.servlet.jsp	2.1.0																	P		O	T	P
46	javax.servlet	javax.servlet.jsp.jstl	1.1.2																				T	P
47	javax.transaction	javax.transaction	1.1.0					T							P	P	P			P	P			P
48	javax.validation	javax.validation	1.0.0.CR3					P																T
49	javax.xml.bind	javax.xml.bind	2.1.7																O					
50	javax.xml.rpc	javax.xml.rpc	1.1.0					T														O		
51	javax.xml.soap	javax.xml.soap	1.3.0																			O		
52	javax.xml.ws	javax.xml.ws	2.1.1					O														O		
53	org.junit	junit	3.8.2																	O				
54	net.sourceforge.jexcelapi	jxl	2.6.6																					O
55	net.sourceforge.cglib	net.sf.cglib	2.2.0	O			O	O													T	T		T
56	net.sourceforge.ehcache	net.sf.ehcache	1.6.2						O															
57	net.sourceforge.jasperreports	net.sf.jasperreports	2.0.5						O															O
58	com.oracle.oc4j	oracle.classloader	10.1.3.1					P																
59	com.oracle.toplink.essentials	oracle.toplink.essentials	2.0.0.b41-beta2															O						
60	org.aopalliance	org.aopalliance	1.0.0	X				X								X	O				X	X		
61	org.apache.axis	org.apache.axis	1.4.0																			O		
62	org.apache.catalina	org.apache.catalina	6.0.16											P	P									
63	org.apache.commons	org.apache.commons.collections	3.2.0						O															
64	org.apache.commons	org.apache.commons.dbcp	1.2.2.osgi					T										T						
65	org.apache.commons	org.apache.commons.fileupload	1.2.0																			O	O	T
66	org.apache.commons	org.apache.commons.httpclient	3.1.0																			O		
67	org.apache.commons	org.apache.commons.logging	1.1.1	X		X	X	X	X	X	X				X	X	X	X	X	X	X	X	X	X
68	org.apache.commons	org.apache.commons.pool	1.4.0	O				T								O								
69	org.apache.derby	org.apache.derby	10.5.1000001.76												O									
70	org.apache.log4j	org.apache.log4j	1.2.15	T			T	T	T	O	T											O		
71	org.apache.openjpa	org.apache.openjpa	1.1.0															O						
72	org.apache.poi	org.apache.poi	3.0.2.FINAL						T															O
80	org.apache.xerces	org.apache.xerces	2.8.1																					
81	org.apache.xmlbeans	org.apache.xmlbeans	2.4.0																					
82	org.aspectj	org.aspectj.weaver	1.6.5.RELEASE	O																			O	
83	org.codehaus.castor	org.castor	1.2.0																					
84	org.codehaus.groovy	org.codehaus.groovy	1.6.3																					
85	org.codehaus.jackson	org.codehaus.jackson.mapper	1.0.0																					
86	org.codehaus.jettison	org.codehaus.jettison	1.0.1																					
87	org.custommonkey.xmlunit	org.custommonkey.xmlunit	1.2.0																					
88	org.dom4j	org.dom4j	1.6.1																					
89	org.easymock	org.easymock	2.5.1	T																				T
90	org.easymock	org.easymock.classextension	2.3.0																					
91	org.eclipse.persistence	org.eclipse.persistence	1.0.1																					
92	org.eclipse.persistence	org.eclipse.persistence.jpa	1.0.1																					
93	com.h2database	org.h2	1.0.71																					
94	org.hibernate	org.hibernate	3.3.1.GA																					
95	org.hibernate	org.hibernate.annotations	3.4.0.GA																					
96	org.hibernate	org.hibernate.ejb	3.4.0.GA																					
97	org.hibernate	org.hibernate.validator	4.0.0.Beta2																					
98	org.hsqldb	org.hsqldb	1.8.0.9																					
99	org.jaxen	org.jaxen	1.1.1																					
100	org.jboss.vfs	org.jboss.virtual	2.1.0.GA																					
101	net.sourceforge.jibx	org.jibx.runtime	1.1.5																					
102	org.jruby	org.jruby	1.2.0																					
103	org.junit	org.junit	4.7.0	T																			T	T
104	org.mortbay.jetty	org.mortbay.jetty.server	6.1.9																					
105	org.mozilla.javascript	org.mozilla.javascript	1.7.0.R2																					
106	org.objectweb.asm	org.objectweb.asm	2.2.3																				O	
107	org.objectweb.asm	org.objectweb.asm.commons	2.2.3																				O	
108	com.opensymphony.quartz	org.quartz	1.6.2																					
109	org.testng	org.testng	5.10.0																					
110	org.xmlpull	org.xmlpull	1.1.3.4-O																					
111	org.slf4j	slf4j.jcl	1.5.3																					
112																								

```xml
<?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:aop="http://www.springframework.org/schema/aop"
    xmlns:tx="http://www.springframework.org/schema/tx"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
        https://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/tx
        https://www.springframework.org/schema/tx/spring-tx.xsd
        http://www.springframework.org/schema/aop
        https://www.springframework.org/schema/aop/spring-aop.xsd">
    <bean id="myDataSource" class="org.apache.commons.dbcp.BasicDataSource" destroy-method="close">
        <property name="driverClassName" value="org.hsqldb.jdbcDriver"/>
        <property name="url" value="jdbc:hsqldb:hsql://localhost:9001"/>
        <property name="username" value="sa"/>
        <property name="password" value=""/>
    </bean>

    <bean id="mySessionFactory" class="org.springframework.orm.hibernate5.LocalSessionFactoryBean">
        <property name="dataSource" ref="myDataSource"/>
        <property name="mappingResources">
            <list>
                <value>product.hbm.xml</value>
            </list>
        </property>
        <property name="hibernateProperties">
            <value>hibernate.dialect=org.hibernate.dialect.HSQLDialect</value>
        </property>
    </bean>

   <bean id="transactionManager"
            class="org.springframework.orm.hibernate5.HibernateTransactionManager">
        <property name="sessionFactory" ref="sessionFactory"/>
   </bean>

  <tx:annotation-driven/>
</beans>
```

유연한 확장

- 스프링 부트에 내장된 디폴트 구성을 커스토마이징 하는 매우 자연스럽고 유연한 방법 제공
- 스프링 부트가 스프링을 사용하는 방식을 이해한다면 언제라도 스프링 부트를 제 거하고 원하는 방식으로 재구성 가능
- 스프링 부트처럼 기술과 구성을 간편하게 제공하는 나만의 모듈 작성

스프링 부트의 이해

스프링 부트를 이용한 개발 방법

- 부트가 결정한 기술과 구성, 디폴트 설정을 수용
- 외부 설정 파일을 이용한 설정 변경 방법을 활용
- 아주 빠르게 개발을 시작할 수 있다
- 하지만...

스프링 부트를 이용한 개발의 오해와 한계

- 애플리케이션 기능 코드만 잘 작성하면 된다
- 스프링을 몰라도 개발을 잘 할 수 있다
- 스프링 부트가 직접적으로 보여주지 않는 것은 몰라도 된다
- 뭔가 기술적인 필요가 생기면 검색을 해서 해결한다

스프링 부트를 이해하게 되면

- 스프링 부트가 스프링의 기술을 어떻게 활용하는지 배우고 응용할 수 있다
- 스프링 부트가 선택한 기술, 자동으로 만들어주는 구성, 디폴트 설정이 어떤 것인지 확인할 수 있다
- 필요할 때 부트의 기본 구성을 수정하거나, 확장할 수 있다
- 나만의 스프링 부트 모듈을 만들어 활용할 수 있다

강의에서 살펴볼 내용

- 컨테이너 (관련 작업이) 없는 독립실행형 애플리케이션은 어떻게 만드는가
- 다양한 기술을 빠르게 적용하고 이를 유연하게 확장하게 해주는 스프링 부트의 기술은 어떤 것이고 어떻게 동작하는가
- 스프링 부트는 스프링을 어떻게 잘 활용하는가!

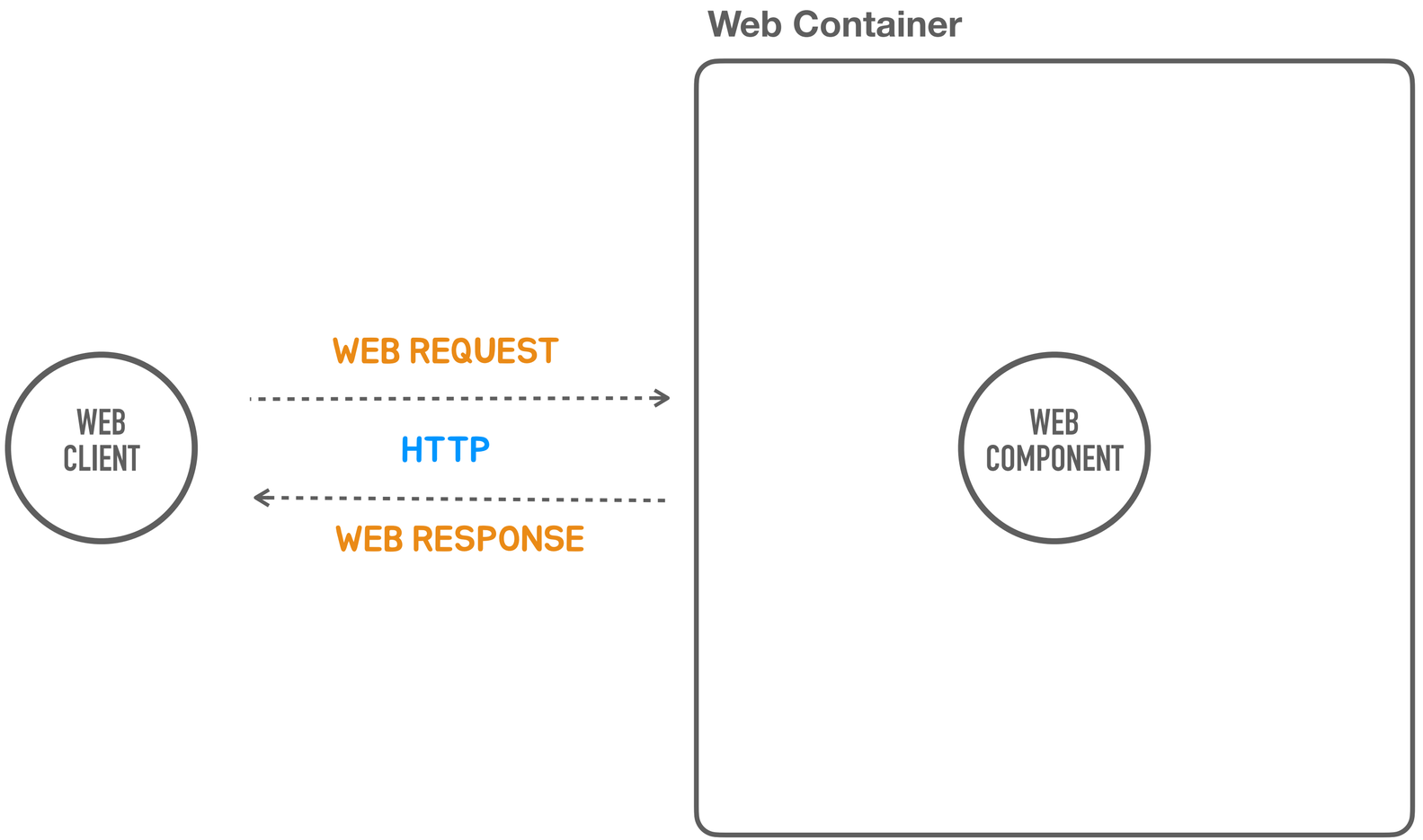
강의의 목표

- 스프링 부트로 만든 스프링 애플리케이션의 기술과 구성 정보를 직접 확인할 수 있다
- 적용 가능한 설정 항목을 파악할 수 있다
- 직접 만든 빈 구성 정보를 적용하고, 그에 따른 변화를 분석할 수 있다
- 스프링 부트의 기술을 꼼꼼히 살펴볼 수 있다

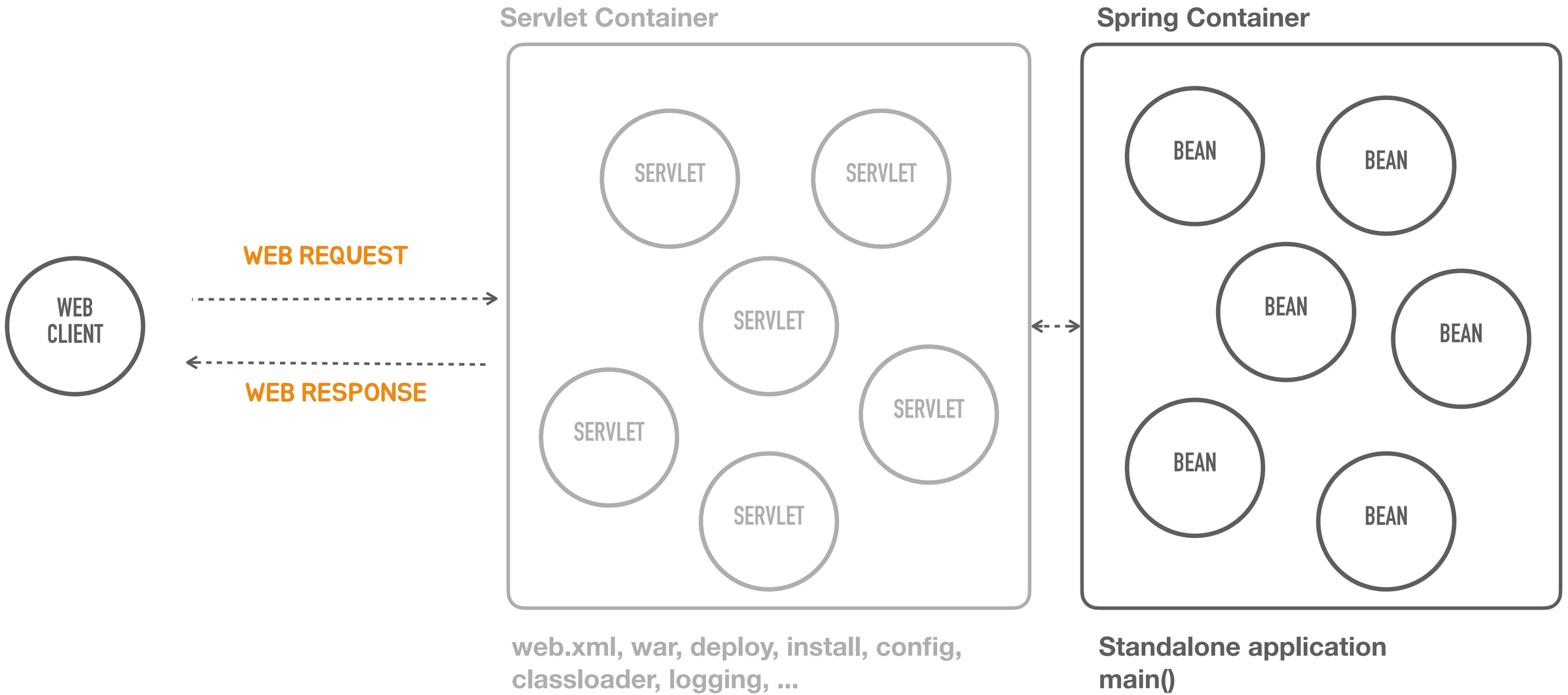
"프레임워크를 효과적으로 재사용하기 위해서는 프레임워크의 최종 모습뿐만 아니라 현재의 모습을 띠게 되기까지 진화한 과정을 살펴 보는 것이 가장 효과적이다. 프레임워크의 진화 과정 속에는 프레임워크의 구성 원리 및 설계 원칙, 재사용 가능한 컨텍스트와 변경 가능성에 관련된 다양한 정보가 들어 있기 때문이다. "

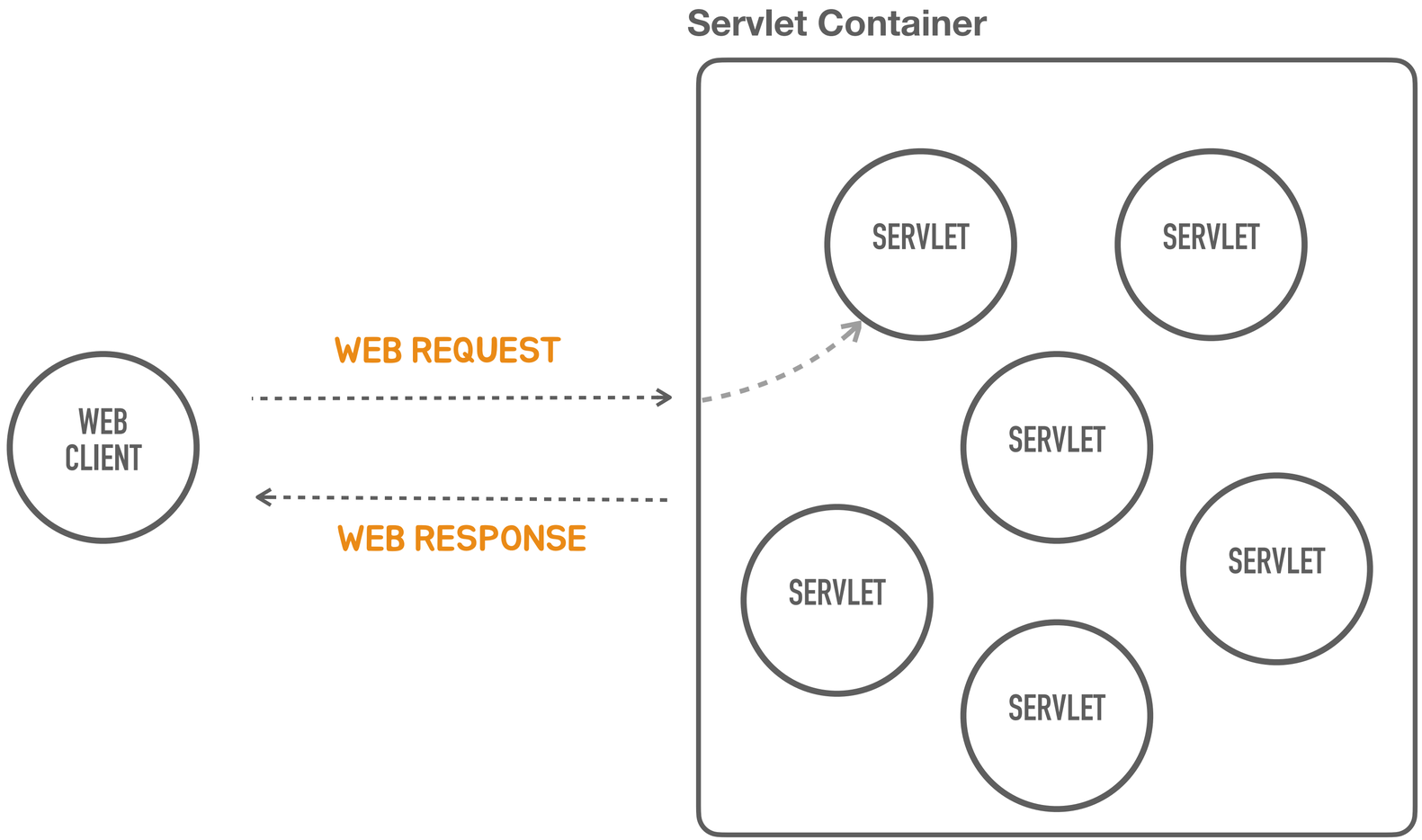
- 조영호 (프레임워크 3부)

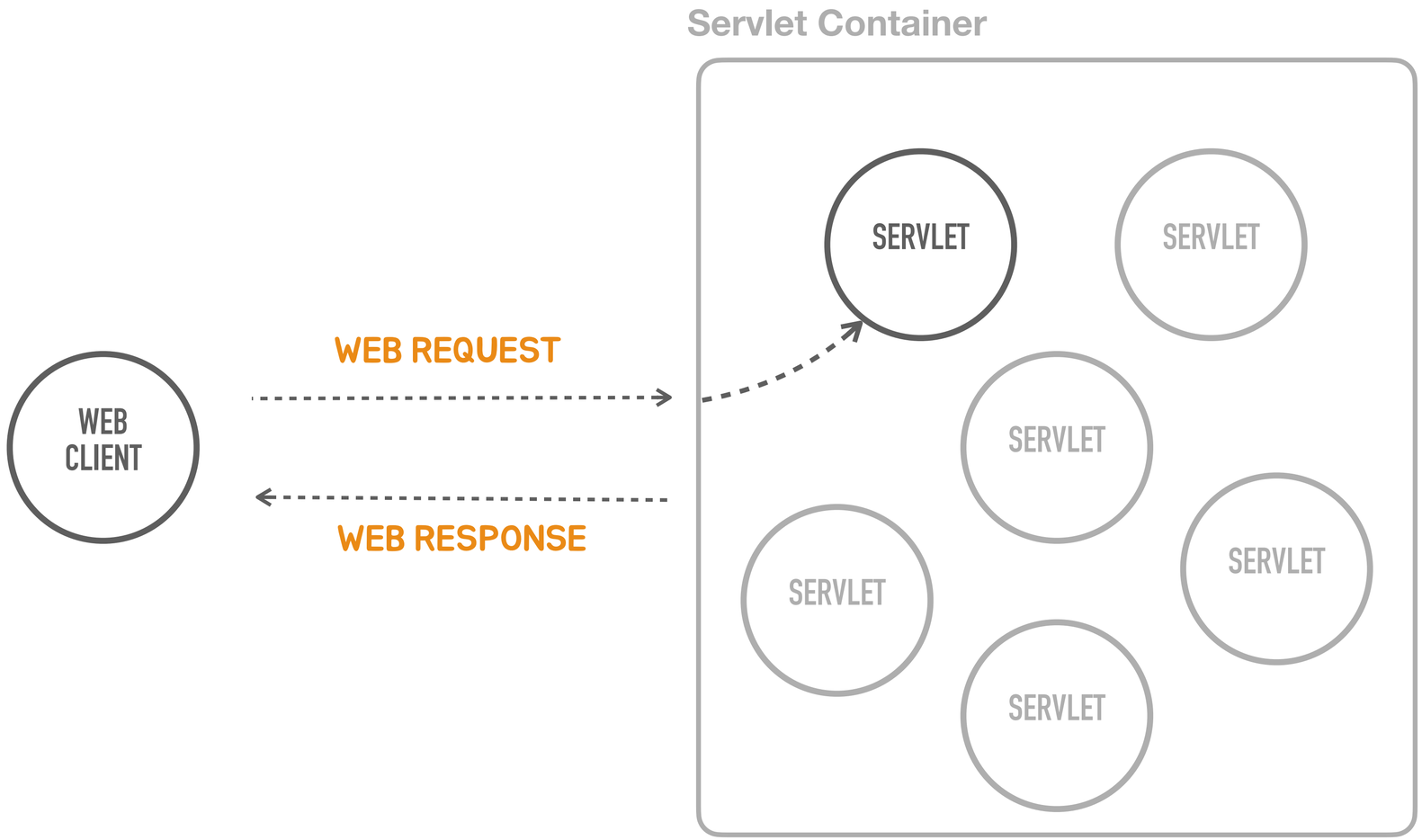
HTTP 요청과 응답

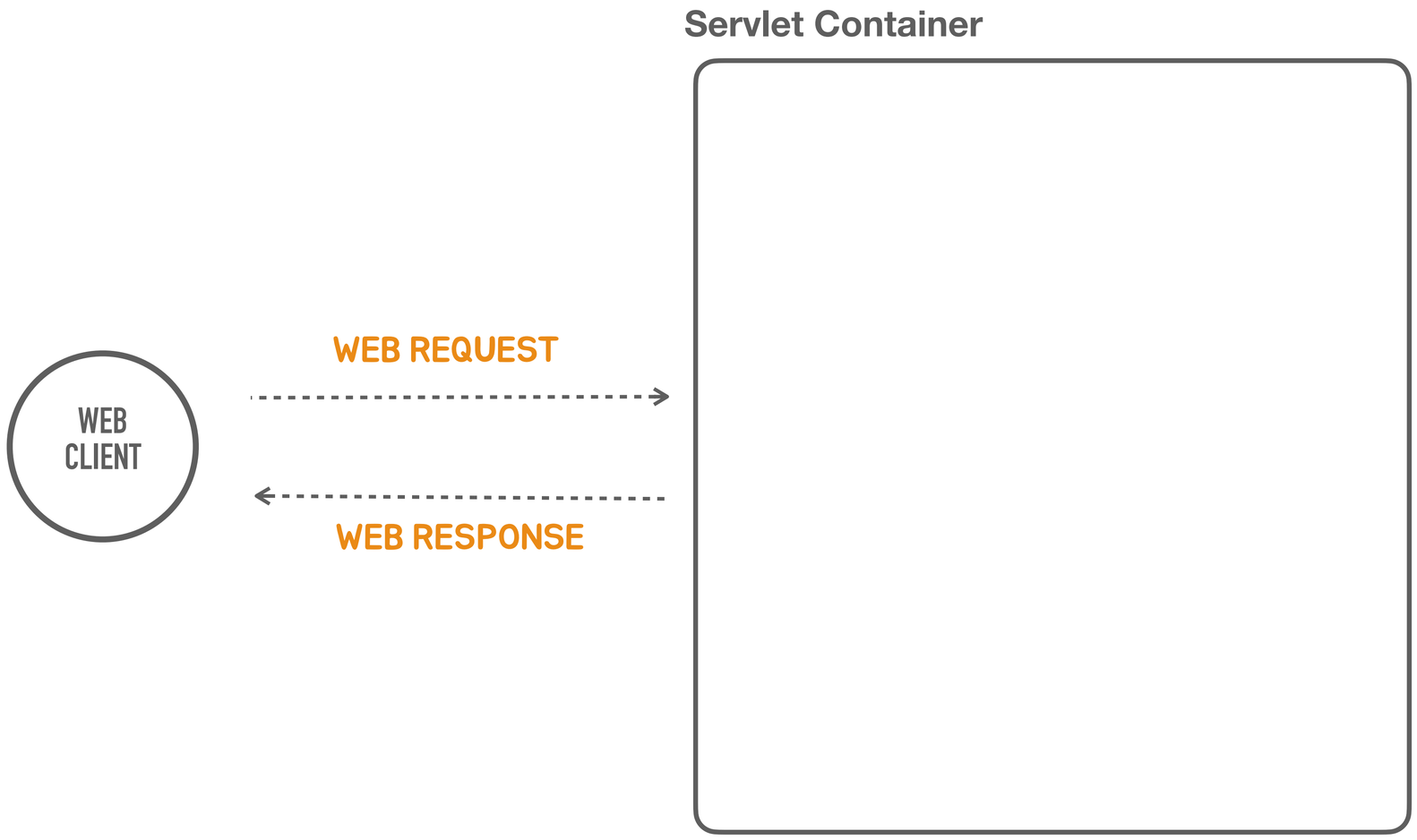

Web Container
WEB
CLIENT
WEB REQUEST
HTTP
WEB RESPONSE
WEB
COMPONENT

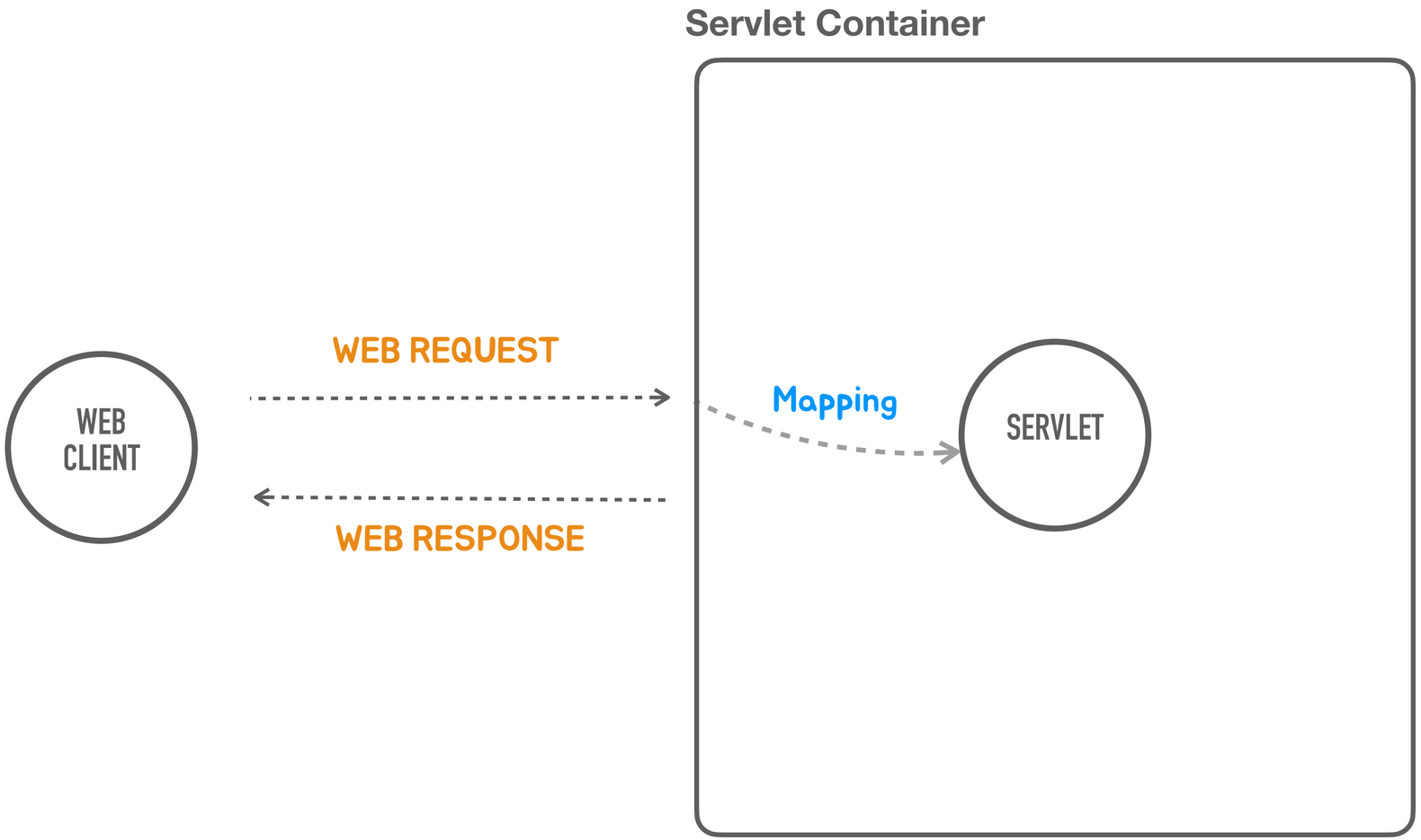
독립 실행형 서블릿 애플리케이션


Servlet Container
Spring Container
WEB CLIENT
WEB REQUEST
WEB RESPONSE
SERVLET
SERVLET
SERVLET
SERVLET
SERVLET
SERVLET
BEAN
BEAN
BEAN
BEAN
BEAN
BEAN
web.xml, war, deploy, install, config, classloader, logging, ...
Standalone application main()


Servlet Container
WEB CLIENT
WEB REQUEST
WEB RESPONSE
SERVLET
SERVLET
SERVLET
SERVLET
SERVLET
SERVLET


Servlet Container
SERVLET
SERVLET
SERVLET
SERVLET
SERVLET
SERVLET
WEB REQUEST
WEB RESPONSE
WEB CLIENT


Servlet Container
WEB REQUEST
WEB CLIENT
WEB RESPONSE

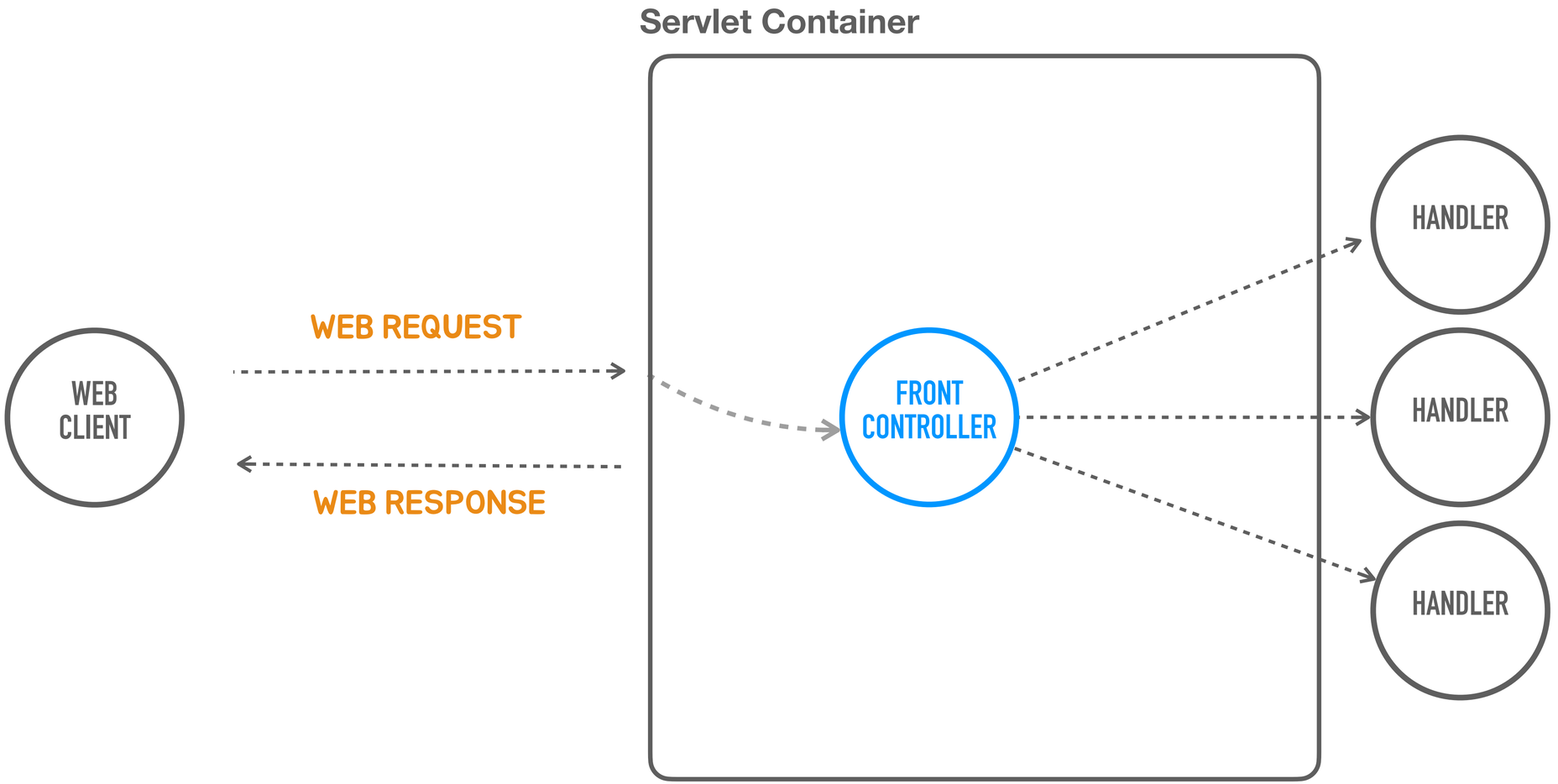

Servlet Container
WEB REQUEST
WEB CLIENT
Mapping
SERVLET
WEB RESPONSE

Request

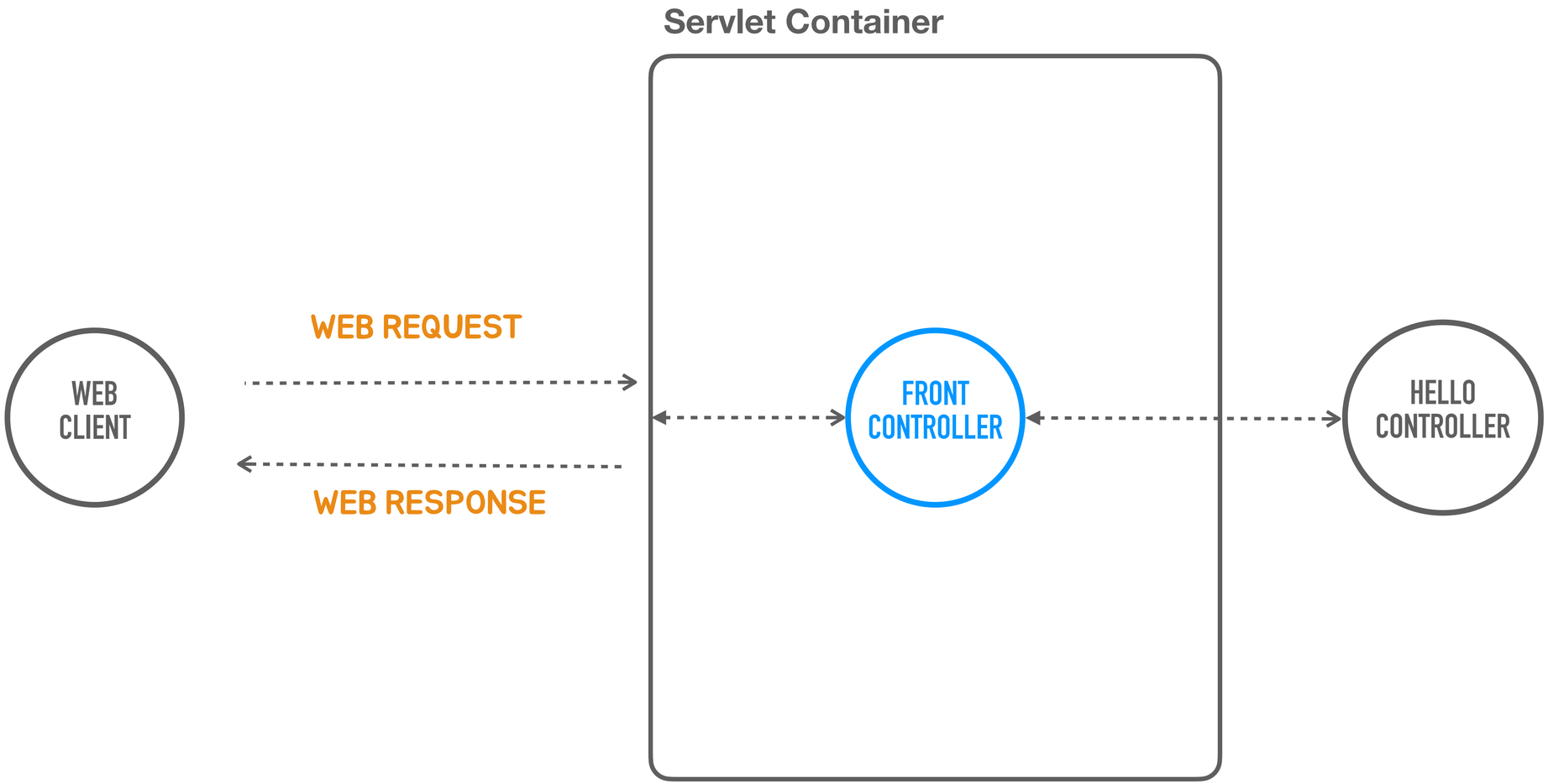
- Request Line: Method, Path, HTTP Version
- Headers
- Message Body

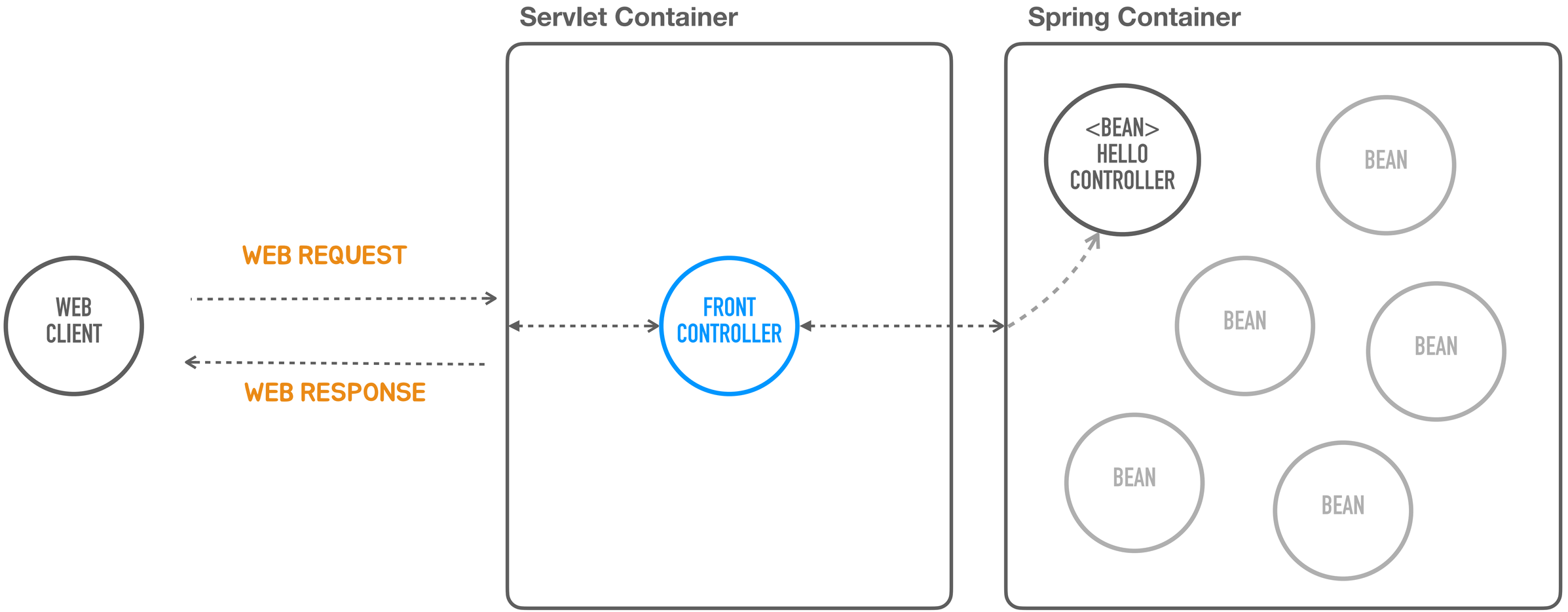
Response

- Status Line: HTTP Version, Status Code, Status Text
- Headers
- Message Body


Servlet Container
WEB CLIENT
WEB REQUEST
WEB RESPONSE
FRONT CONTROLLER
HANDLER
HANDLER
HANDLER

독립 실행형 스프링 애플리케이션


Servlet Container
WEB REQUEST
WEB CLIENT
WEB RESPONSE
FRONT CONTROLLER
HELLO CONTROLLER


Servlet Container
Spring Container
WEB CLIENT
WEB REQUEST
WEB RESPONSE
FRONT CONTROLLER
<BEAN>
HELLO
CONTROLLER
BEAN
BEAN
BEAN
BEAN
BEAN

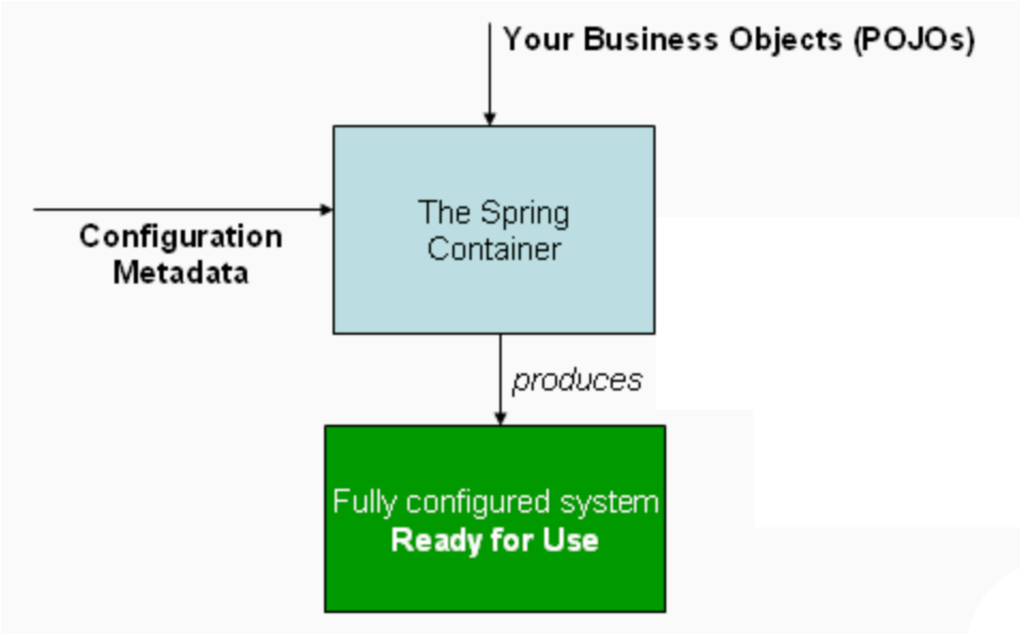
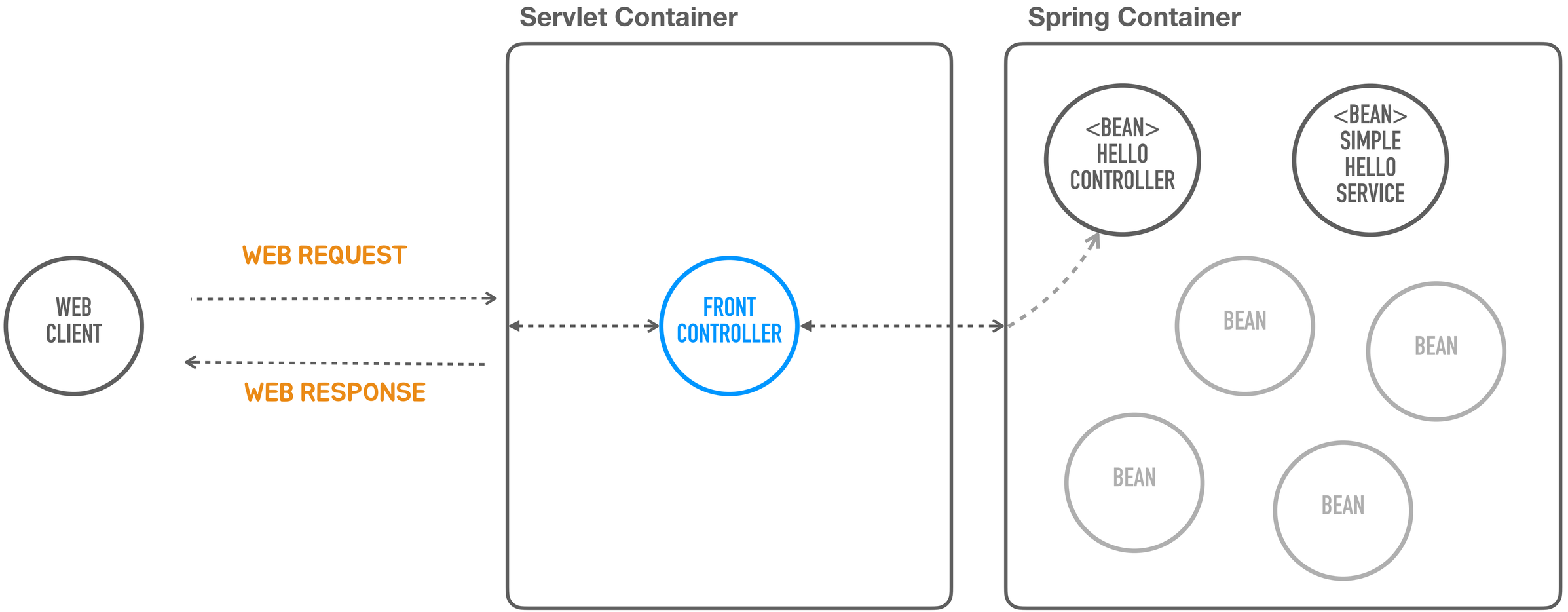
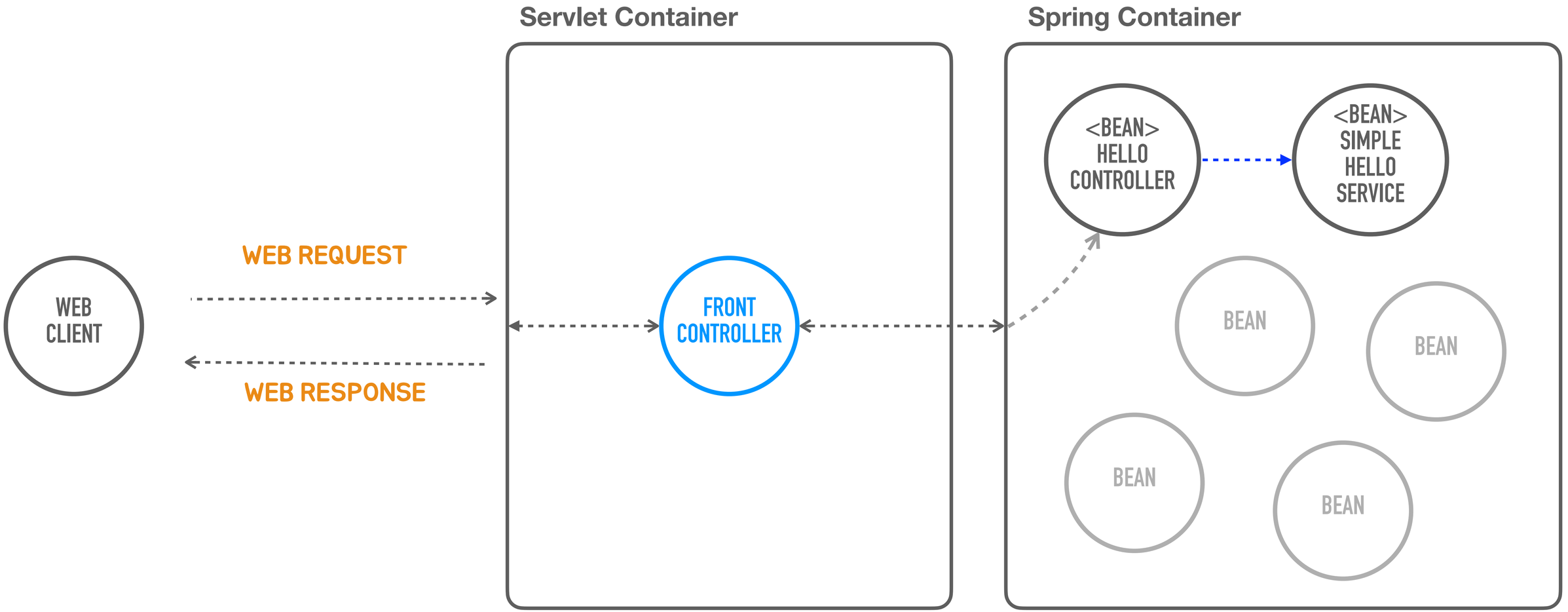
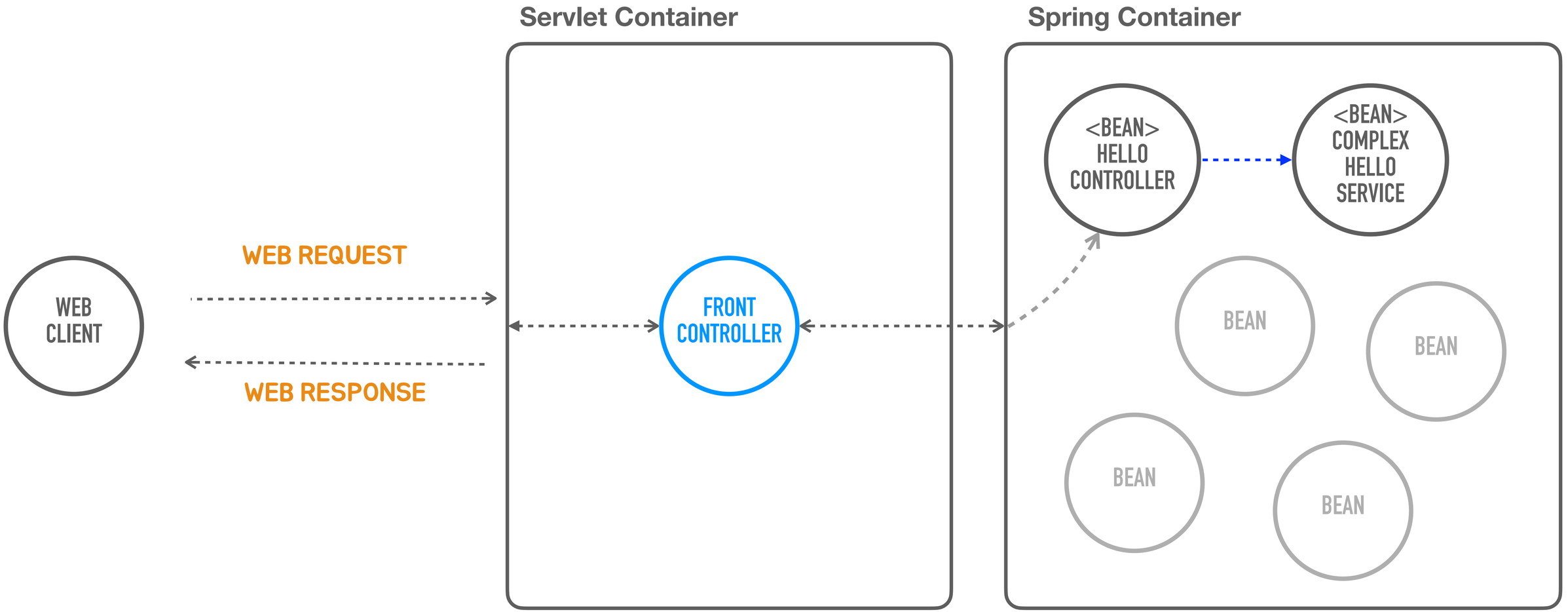


Figure 1. The Spring IoC container

https://docs.spring.io/spring-framework/docs/5.3.x/reference/html/core.html#spring-core


Servlet Container
Spring Container
WEB CLIENT
WEB REQUEST
WEB RESPONSE
FRONT CONTROLLER
<BEAN> HELLO CONTROLLER
<BEAN> SIMPLE HELLO SERVICE
BEAN
BEAN
BEAN
BEAN

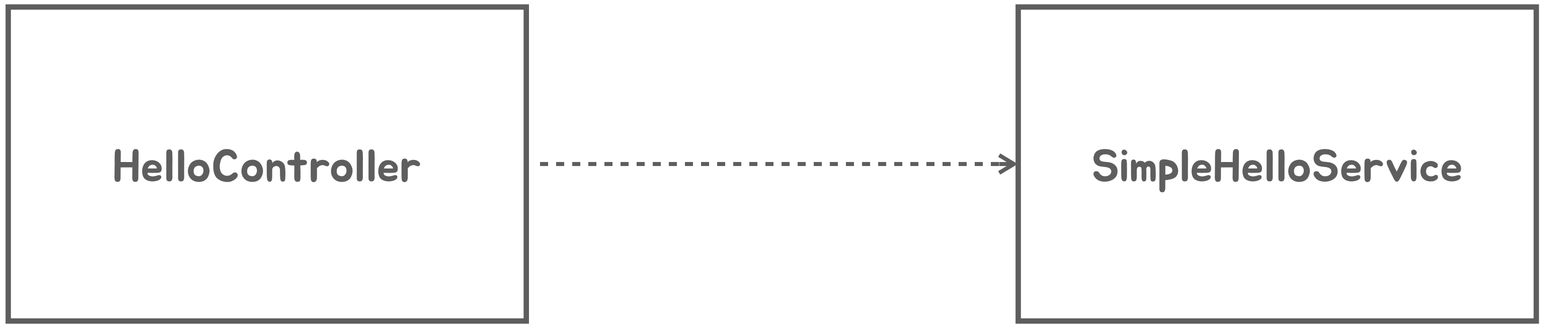

Servlet Container
Spring Container
WEB CLIENT
WEB REQUEST
WEB RESPONSE
FRONT CONTROLLER
<BEAN> HELLO CONTROLLER
<BEAN> SIMPLE HELLO SERVICE
BEAN
BEAN
BEAN
BEAN

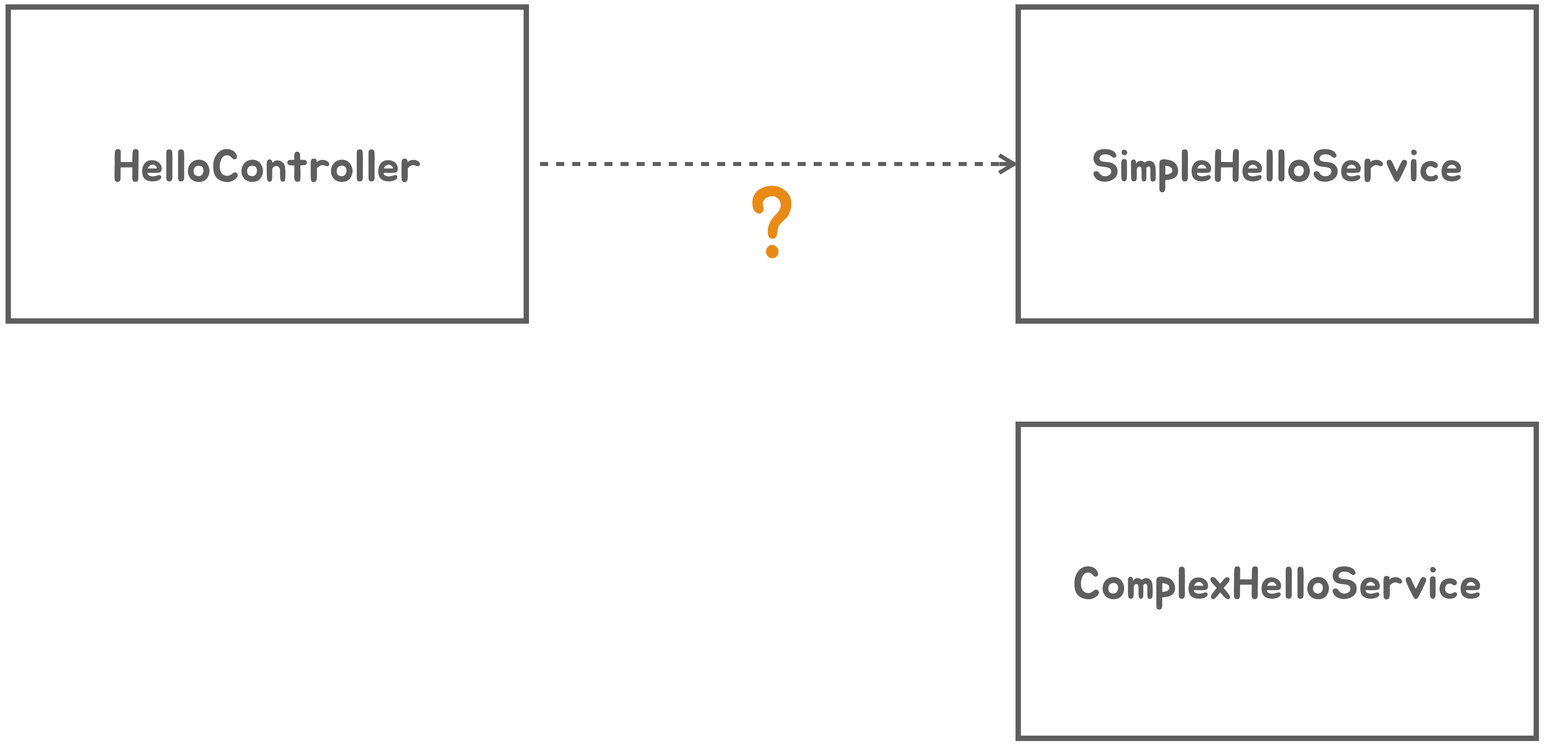

Servlet Container
Spring Container
WEB CLIENT
WEB REQUEST
WEB RESPONSE
FRONT CONTROLLER
<BEAN> HELLO CONTROLLER
<BEAN> COMPLEX HELLO SERVICE
BEAN
BEAN
BEAN
BEAN

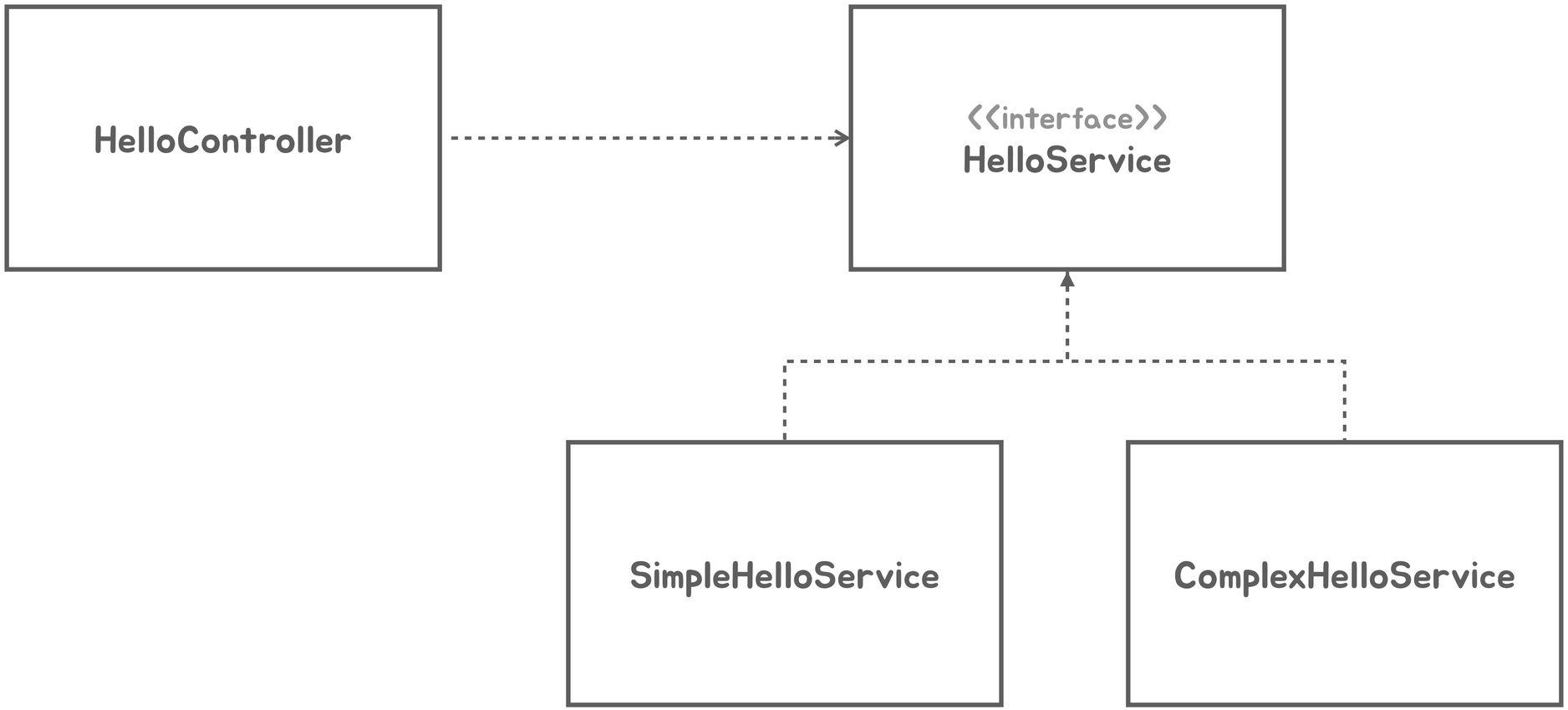
Dependency Injection

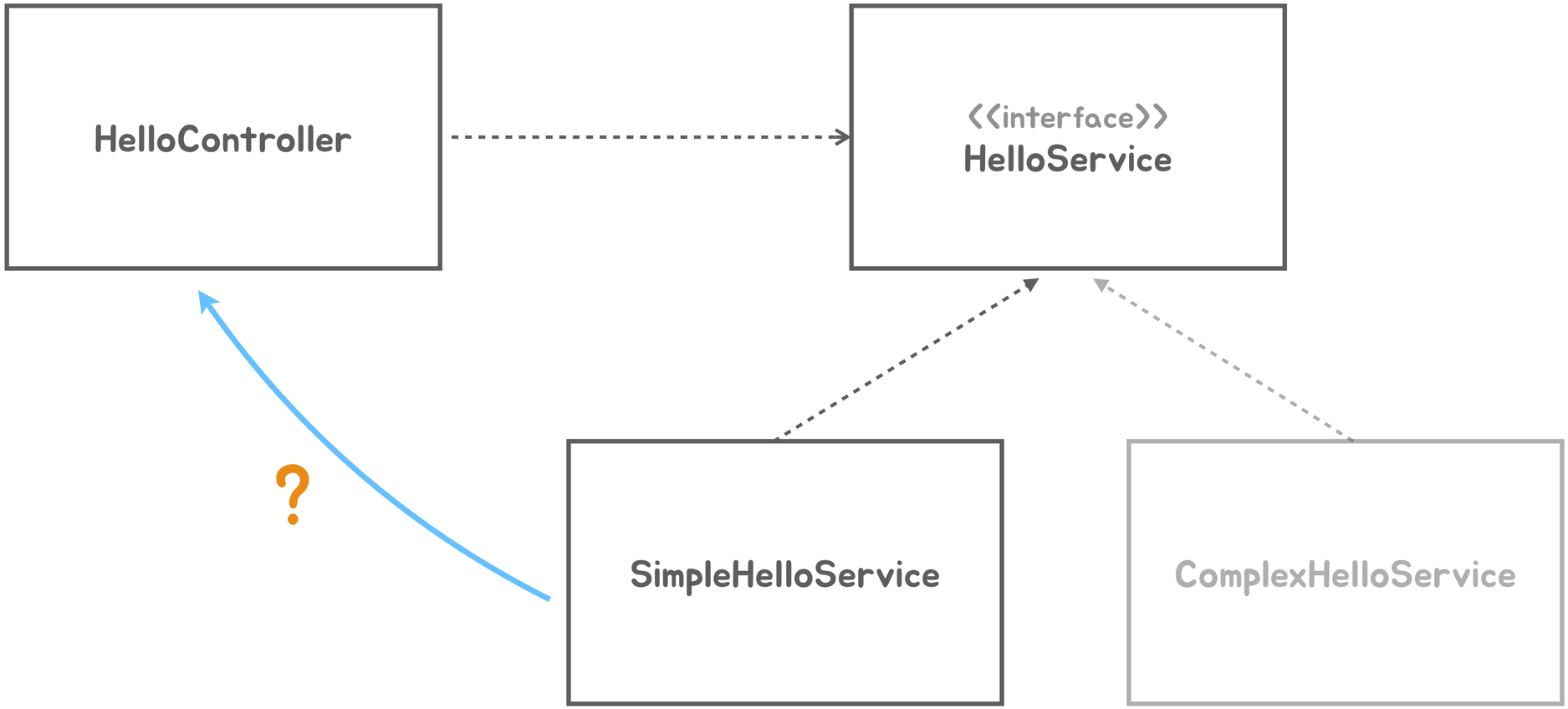
Spring IoC/DI Container

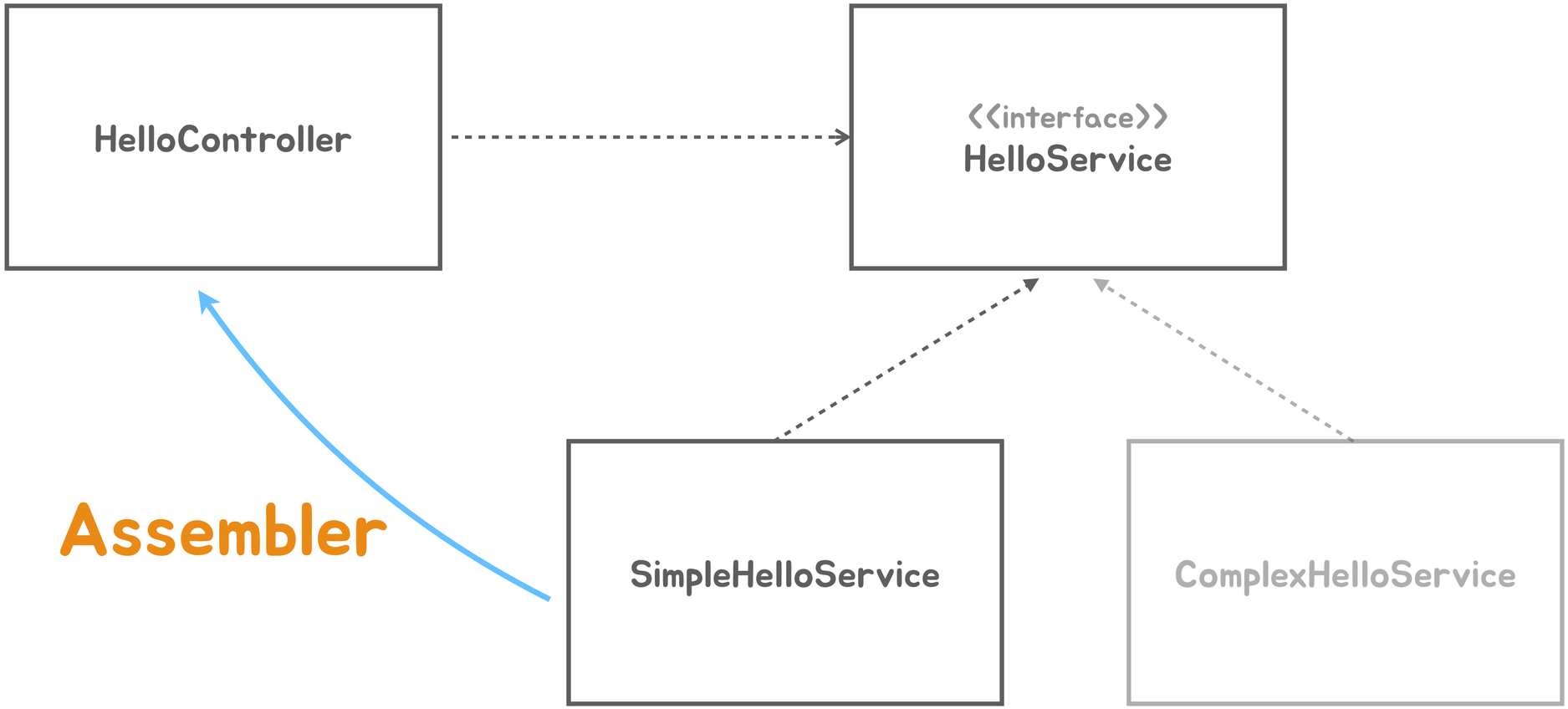

HelloController
SimpleHelloService

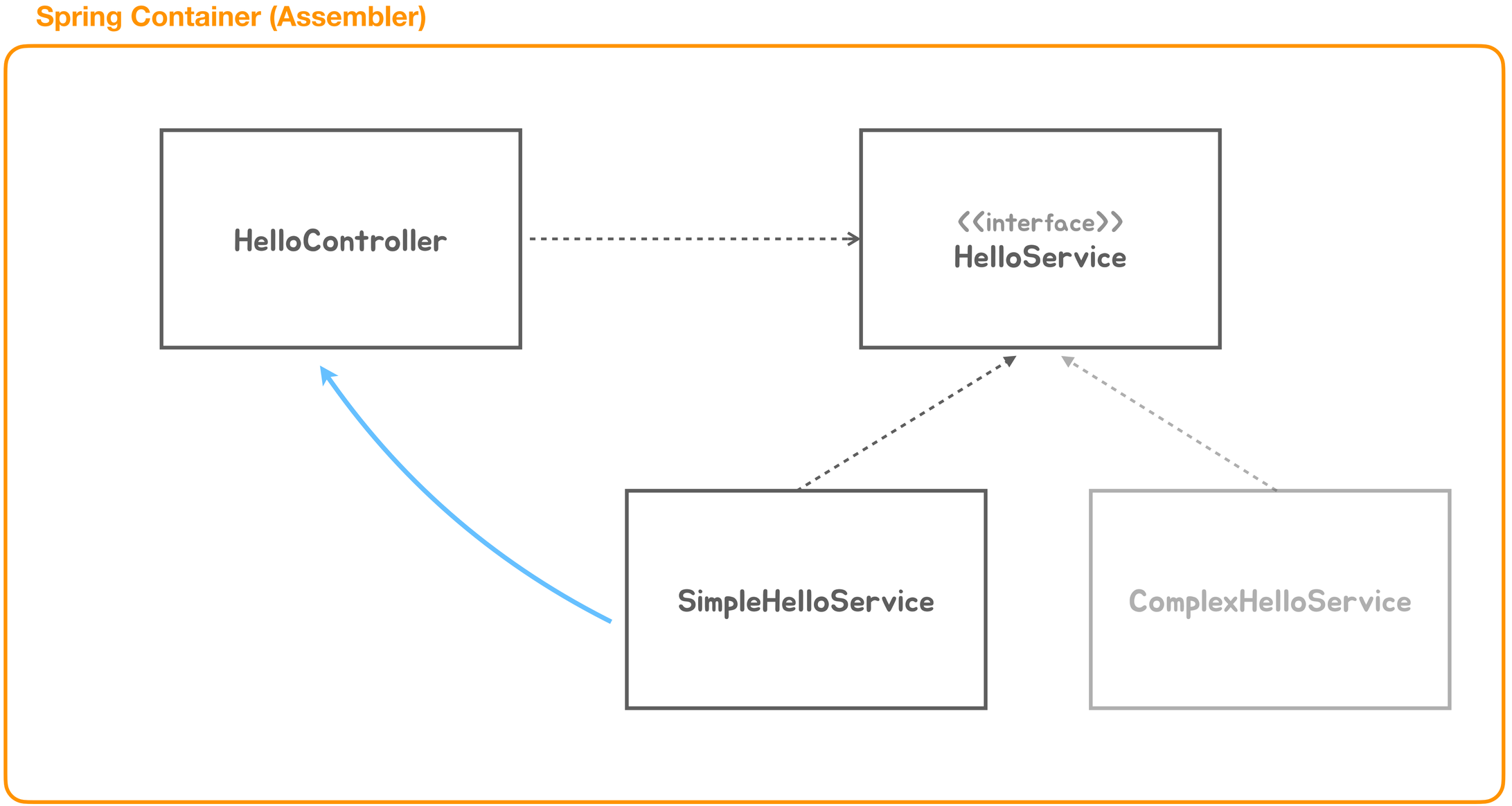

HelloController
SimpleHelloService


HelloController
?
SimpleHelloService
ComplexHelloService

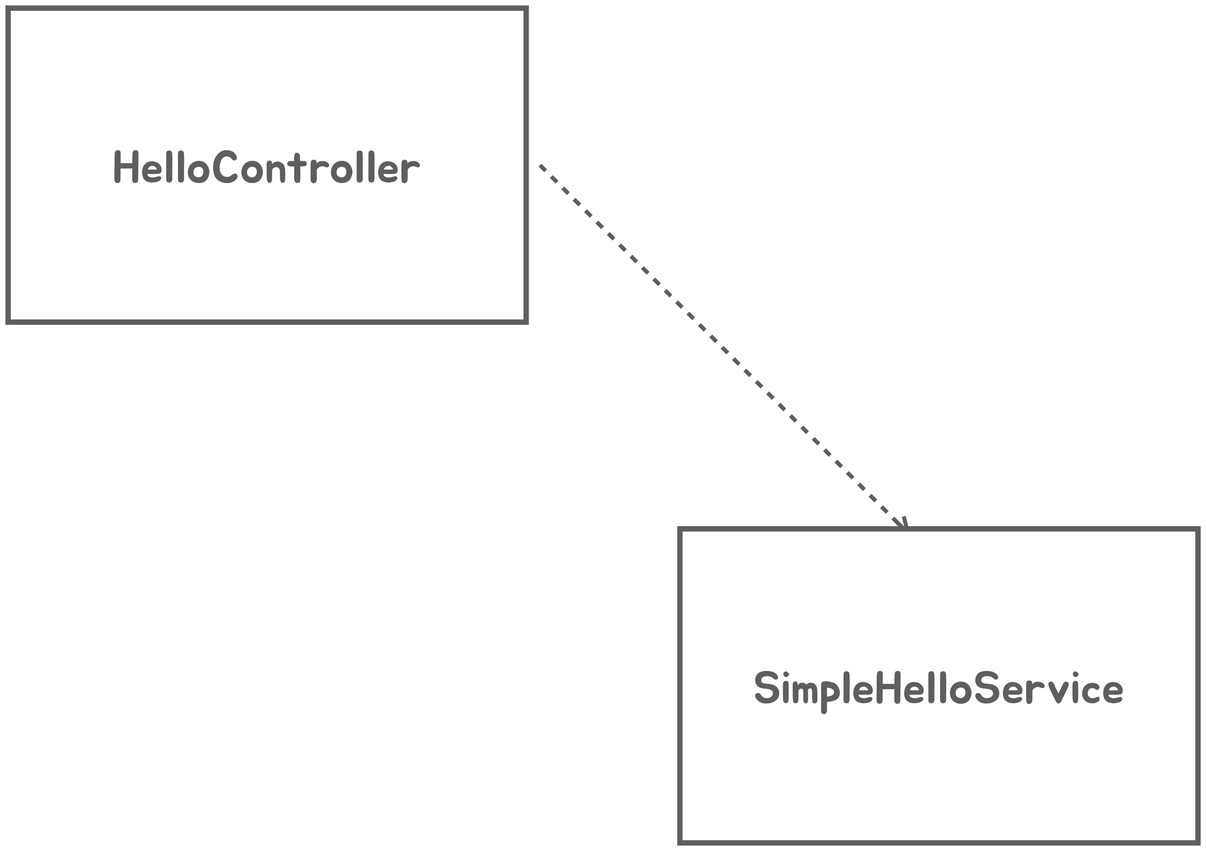

HelloController
<<interface>>
HelloService
SimpleHelloService
ComplexHelloService

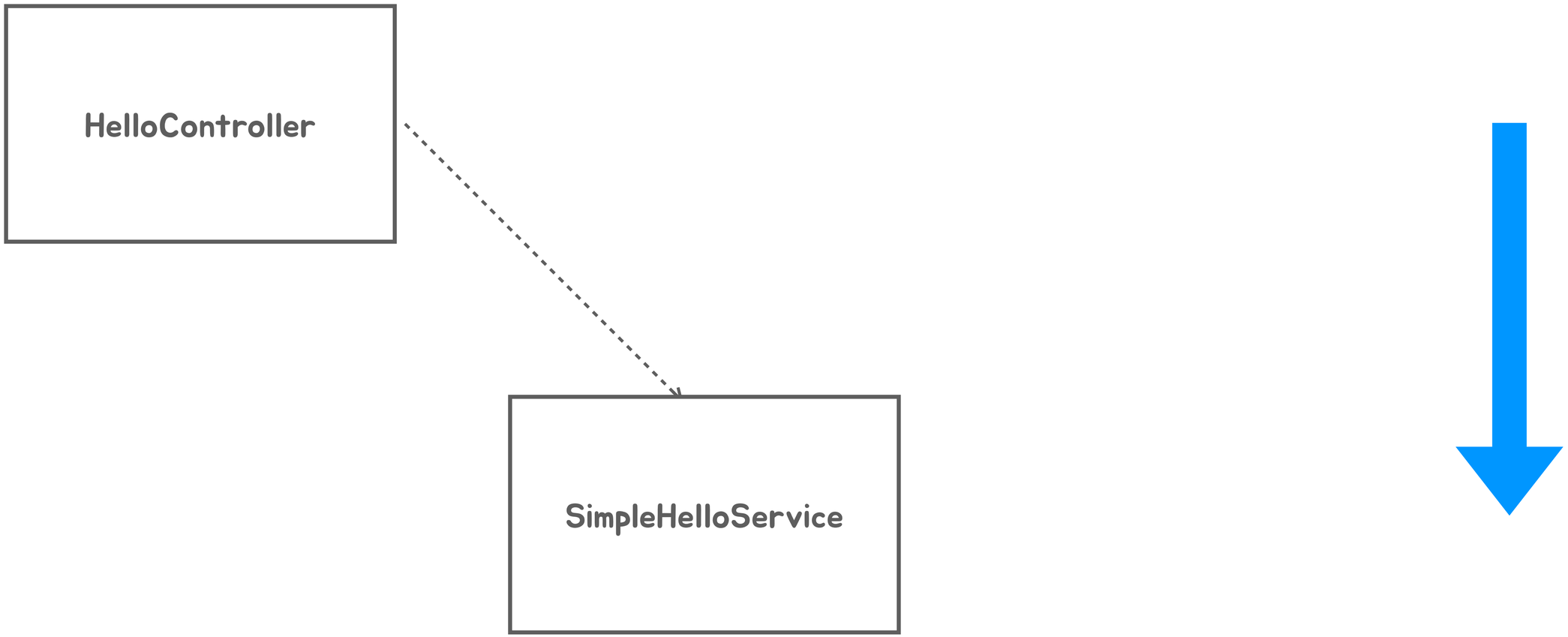

HelloController
<<interface>>
HelloService
?
SimpleHelloService
ComplexHelloService

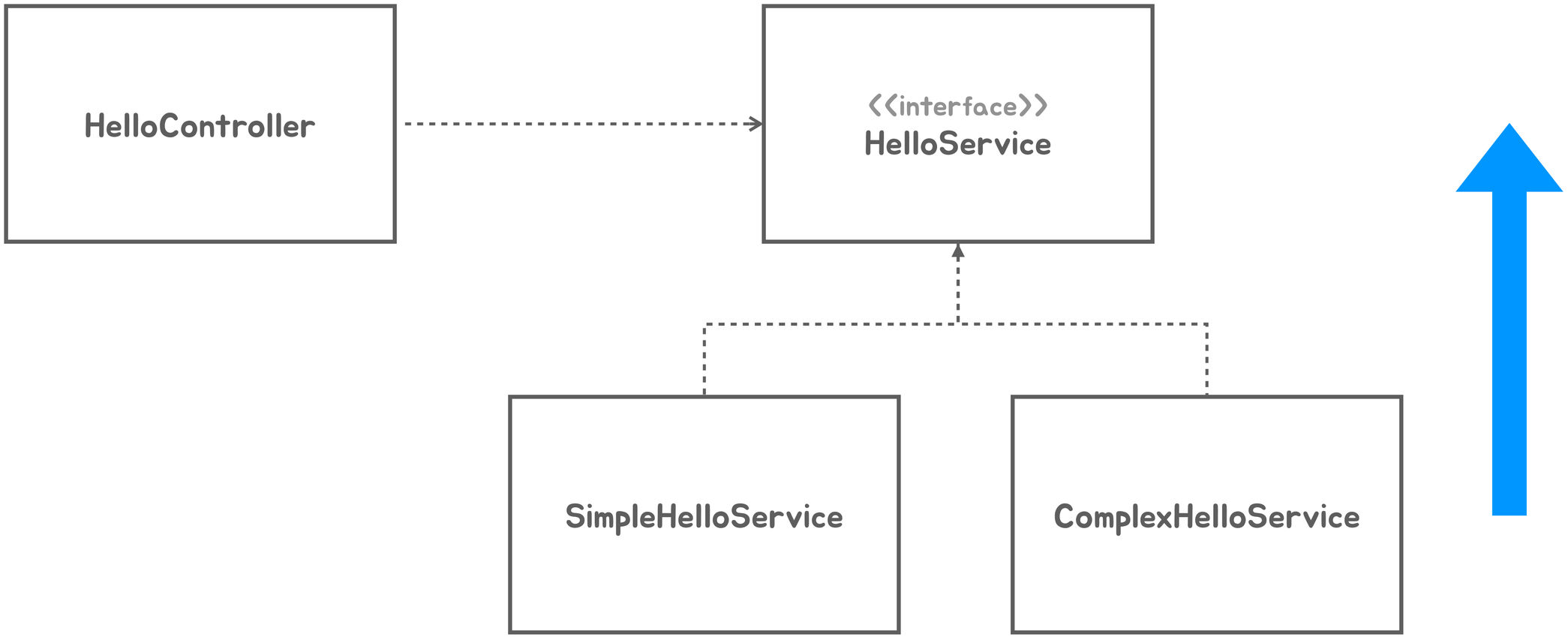

HelloController
<<interface>>
HelloService
Assembler
SimpleHelloService
ComplexHelloService


Spring Container (Assembler)
HelloController
<<interface>>
HelloService
SimpleHelloService
ComplexHelloService

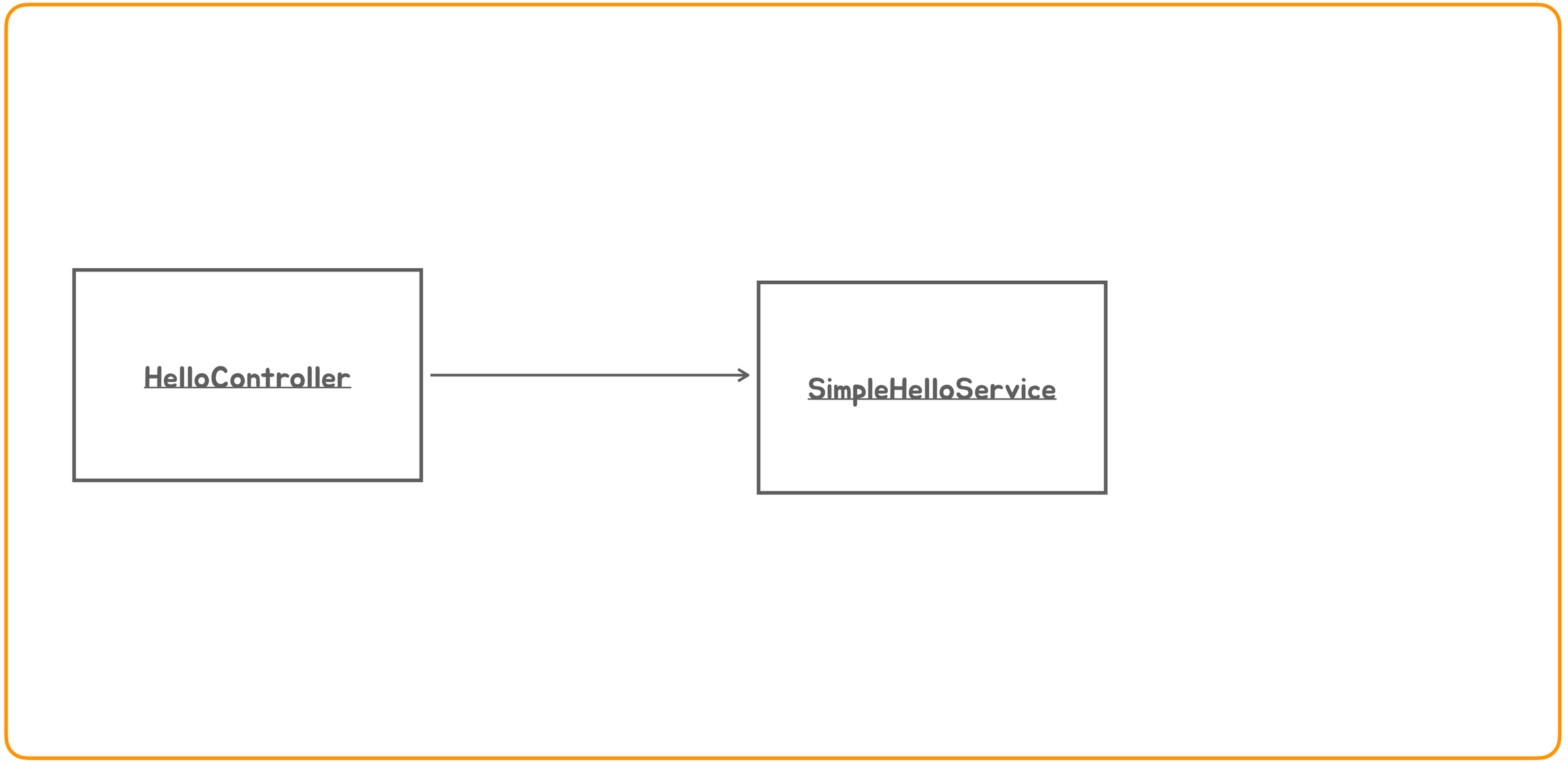
DI와 테스트, 디자인 패턴


HelloController
SimpleHelloService

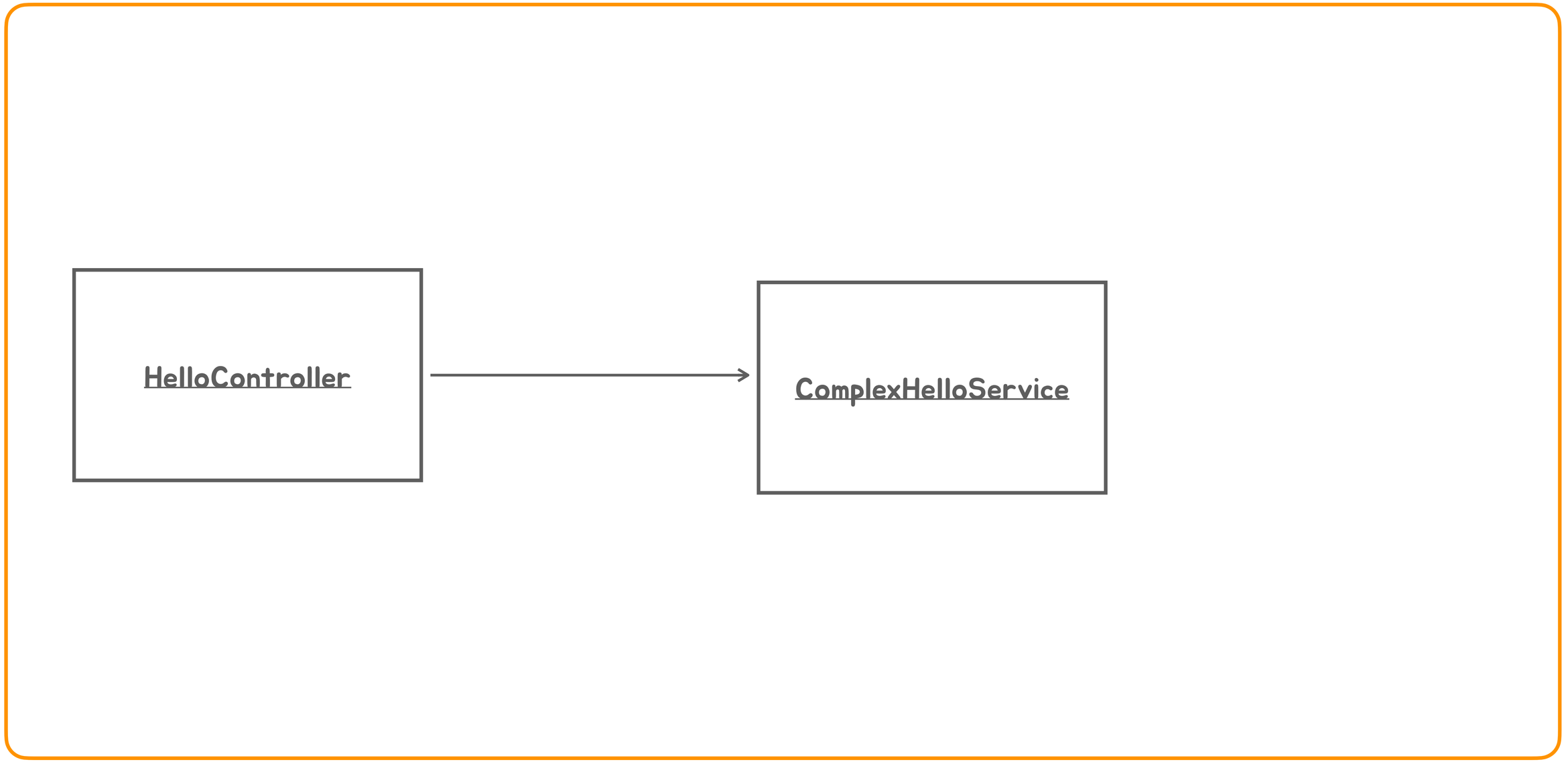

HelloController
SimpleHelloService

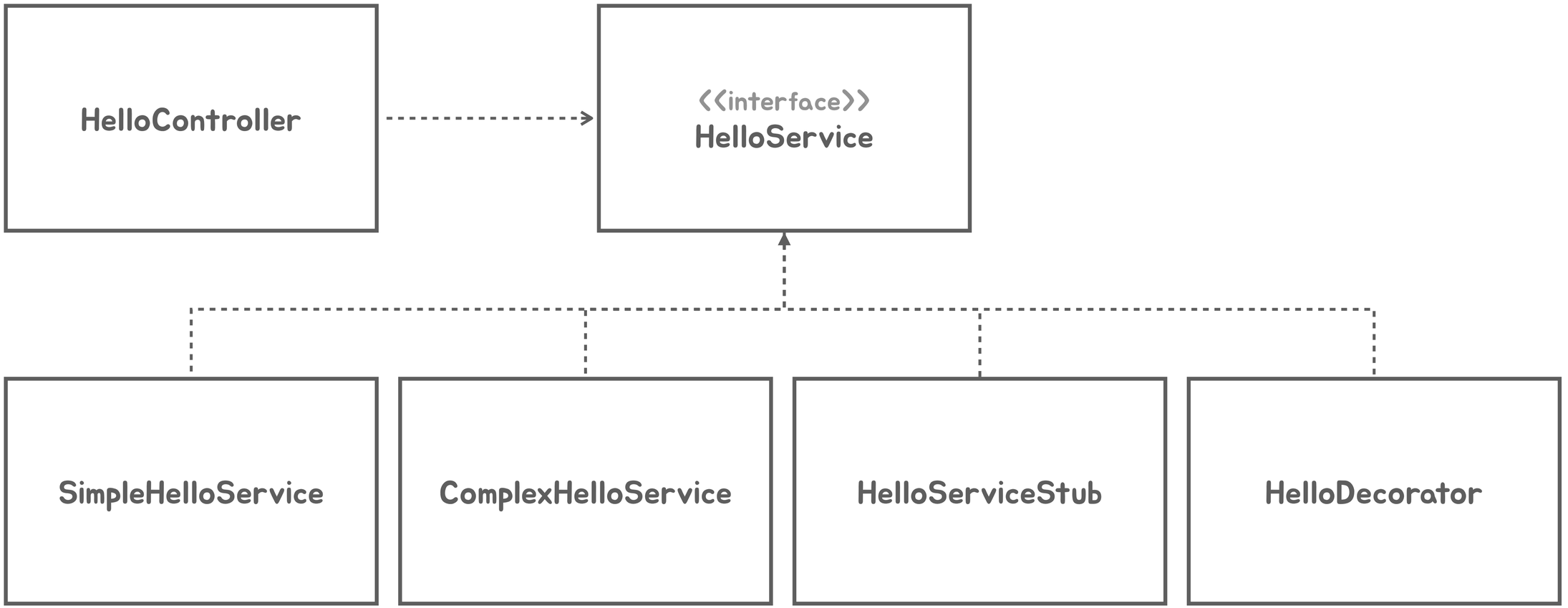

HelloController
<<interface>>
HelloService
SimpleHelloService
ComplexHelloService

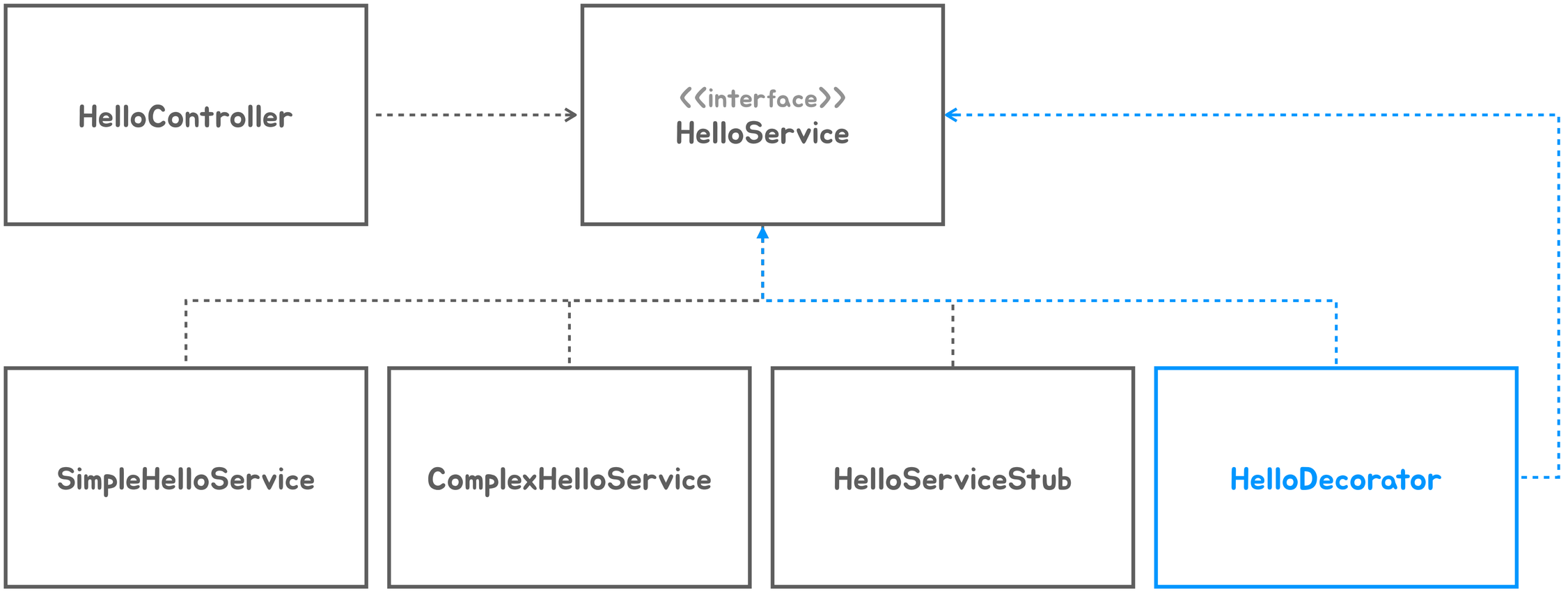
Spring Container (Assembler)



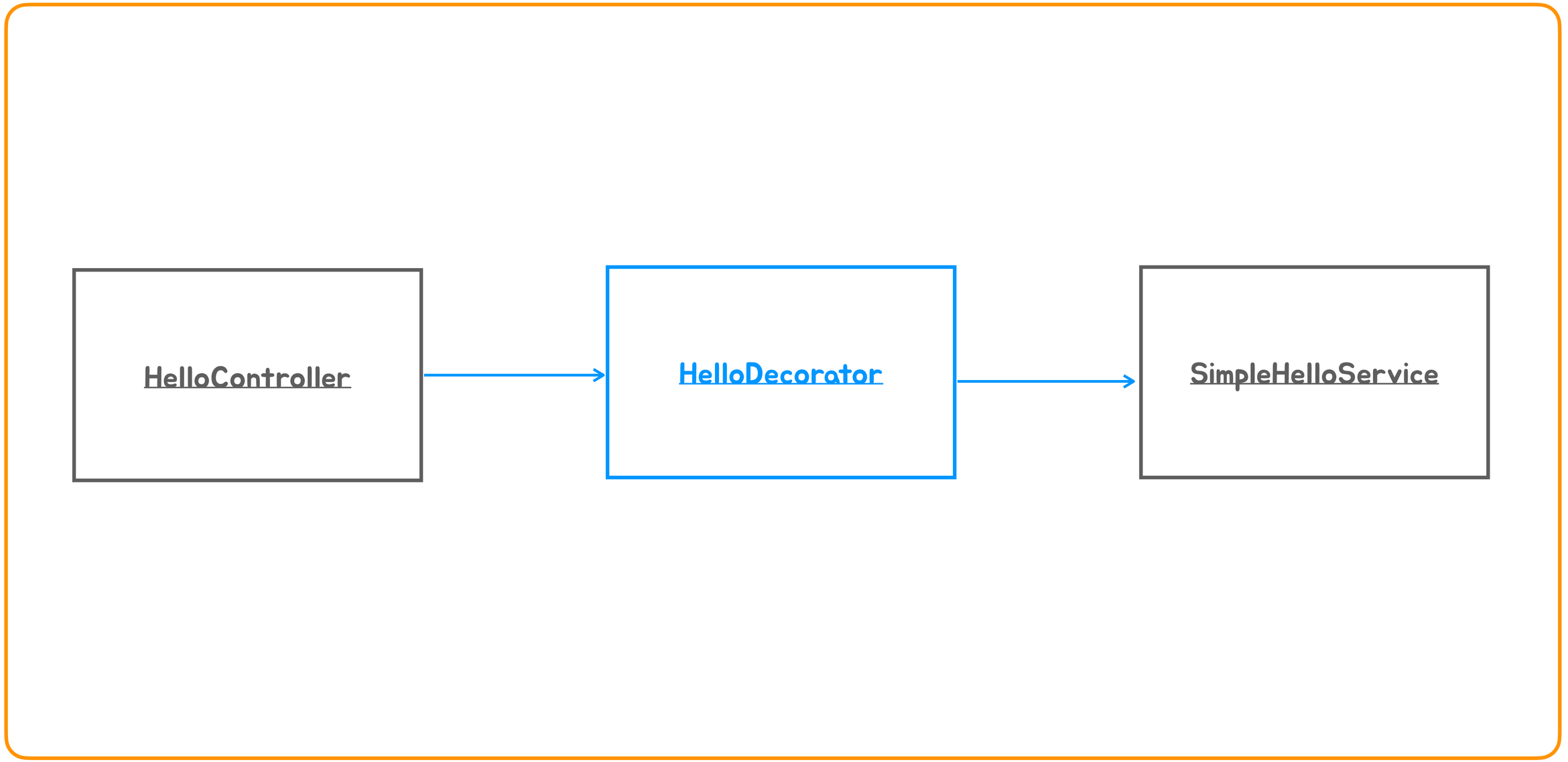
Spring Container (Assembler)

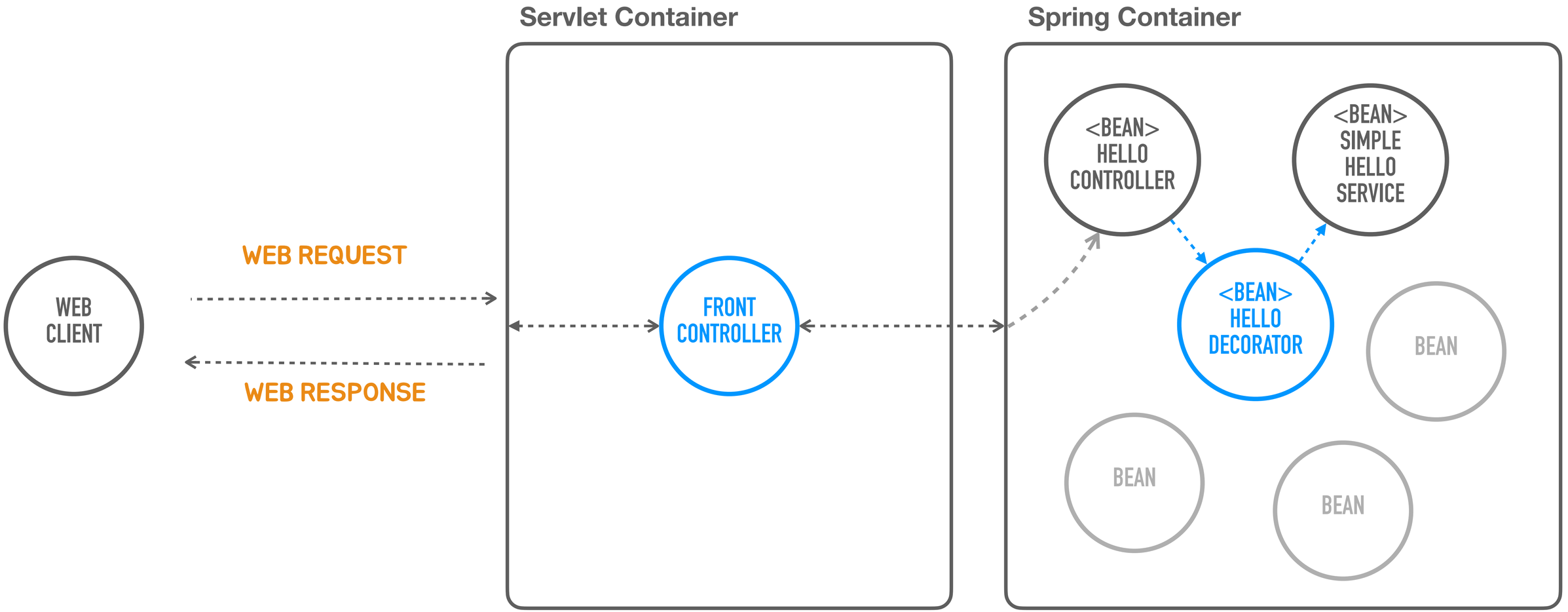



HelloController
<<interface>>
HelloService
SimpleHelloService
ComplexHelloService
HelloServiceStub
HelloDecorator

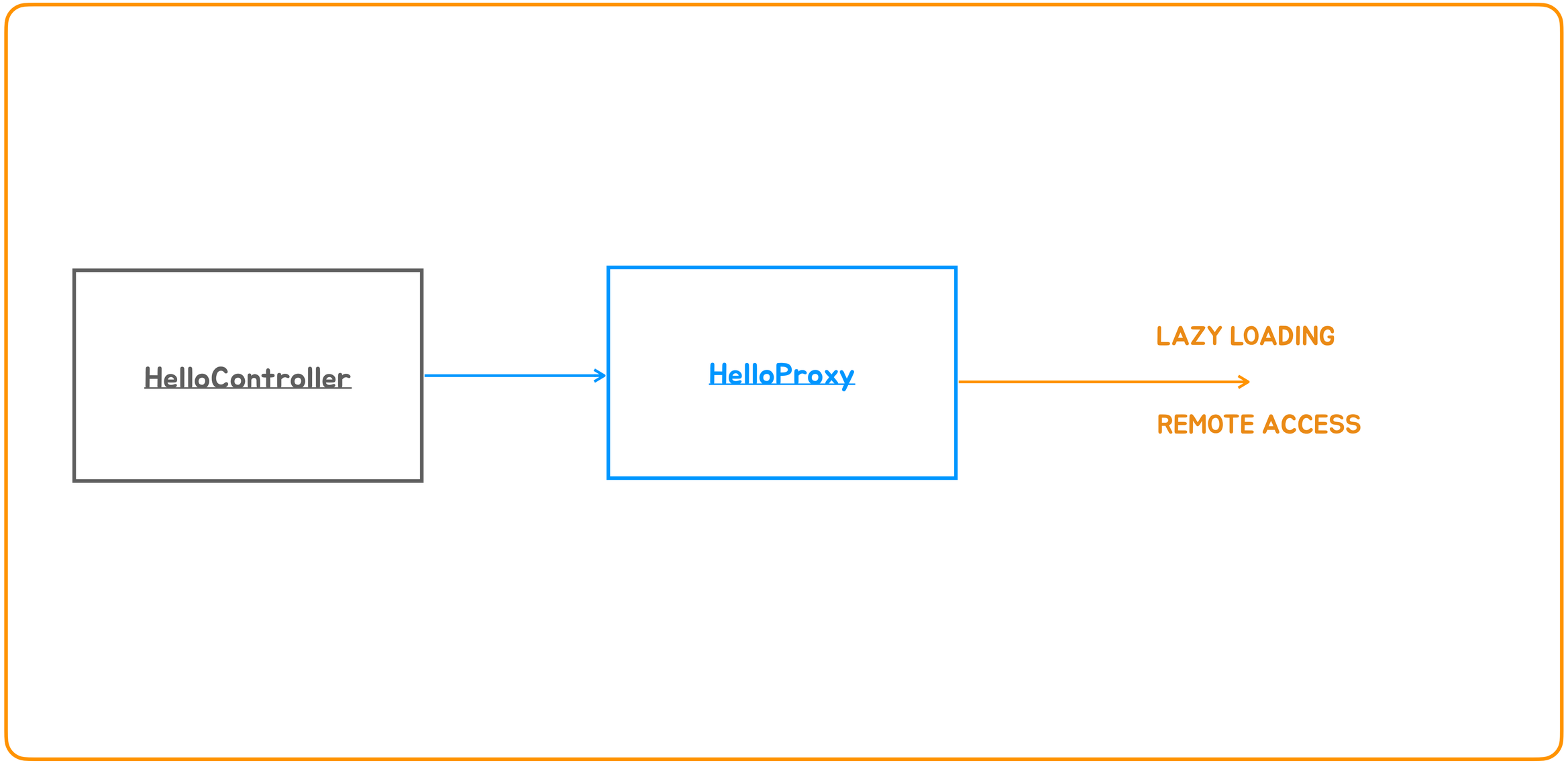

HelloController
<<interface>>
HelloService
SimpleHelloService
ComplexHelloService
HelloServiceStub
HelloDecorator

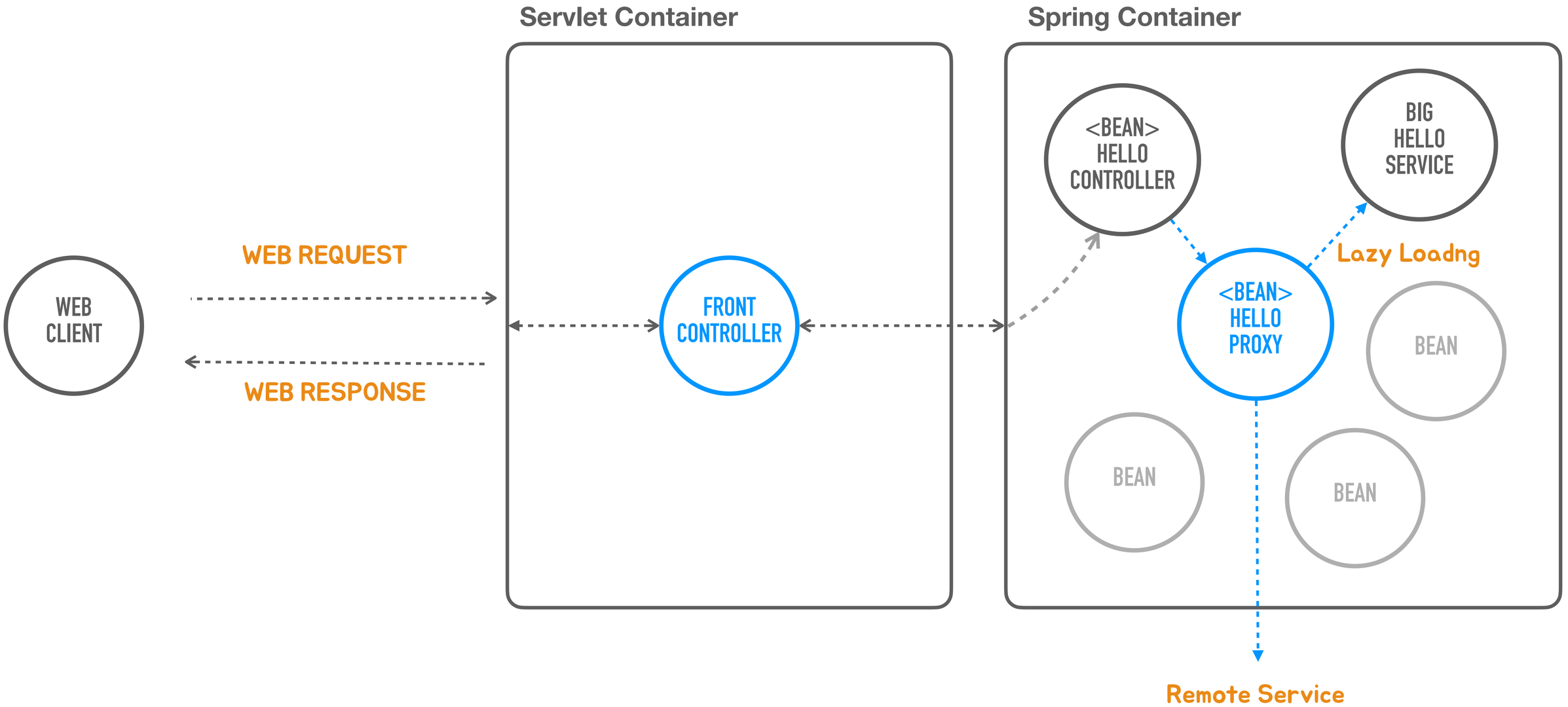
Spring Container (Assembler)




Servlet Container
Spring Container
WEB CLIENT
WEB REQUEST
WEB RESPONSE
FRONT CONTROLLER
<BEAN> HELLO CONTROLLER
<BEAN> SIMPLE HELLO SERVICE
<BEAN> HELLO DECORATOR
BEAN
BEAN
BEAN

Spring Container (Assembler)

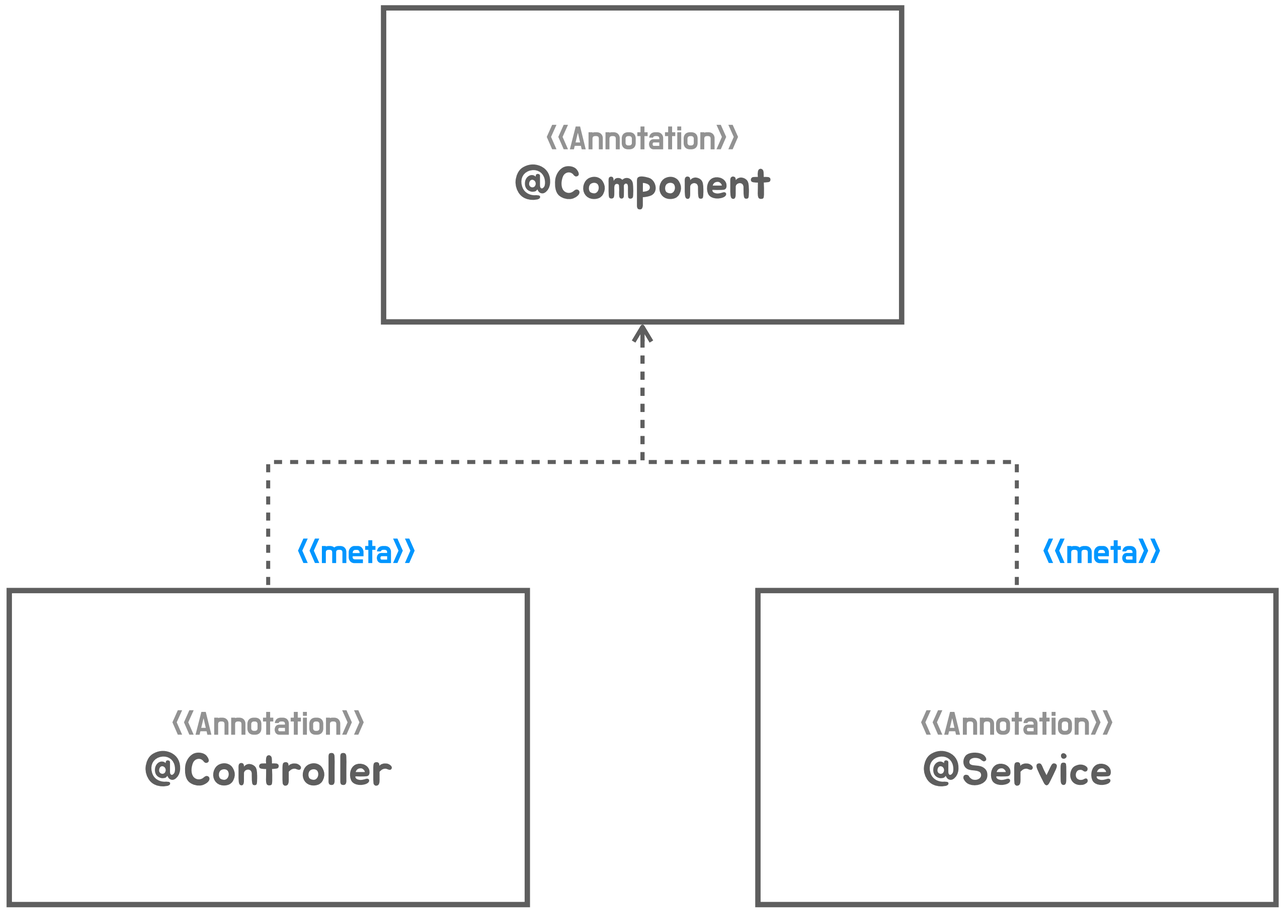


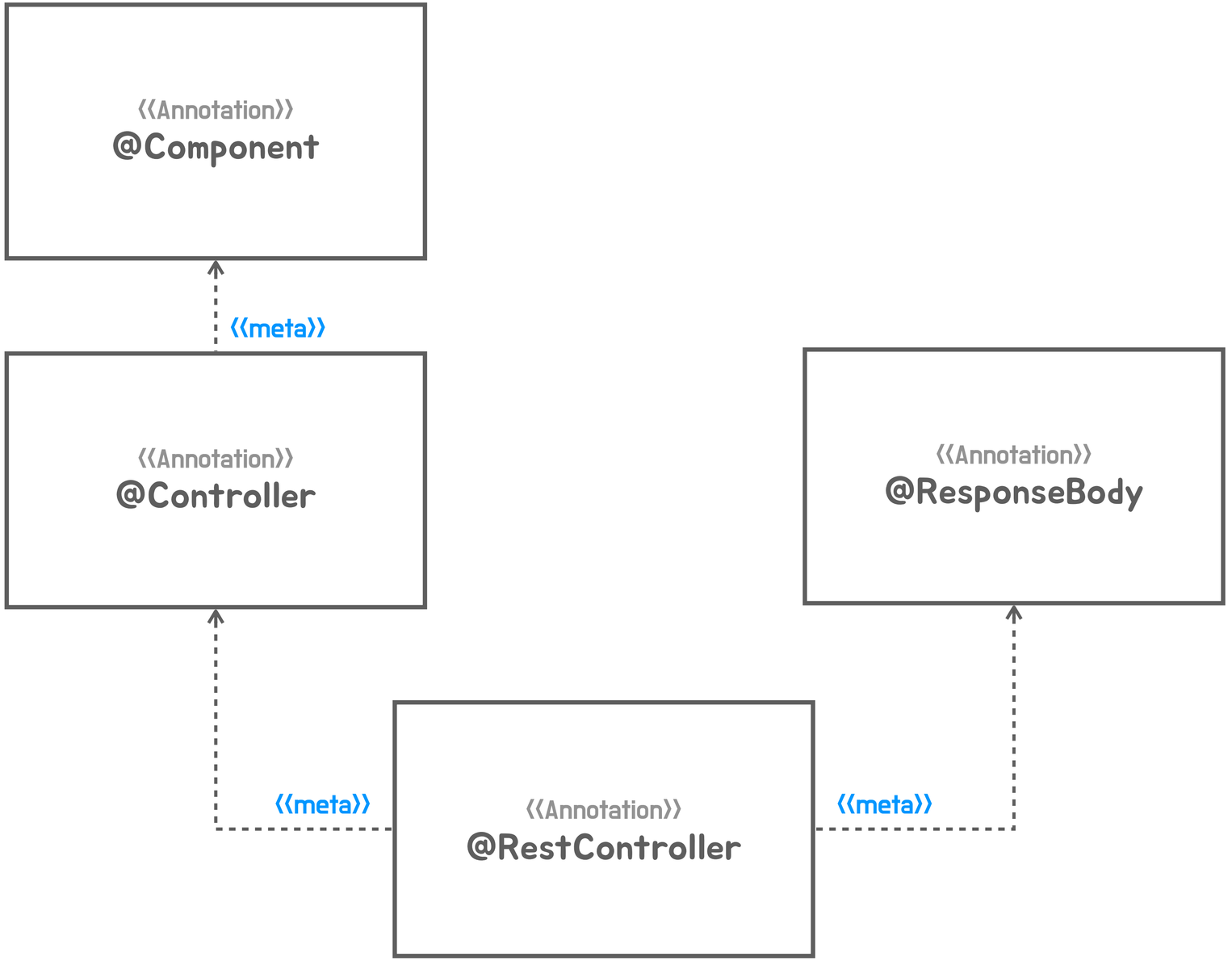

Servlet Container
Spring Container
WEB CLIENT
WEB REQUEST
WEB RESPONSE
FRONT CONTROLLER
<BEAN> HELLO CONTROLLER
BIG HELLO SERVICE
Lazy Loadng
<BEAN> HELLO PROXY
BEAN
BEAN
BEAN
Remote Service

자동 구성 기반 애플리케이션

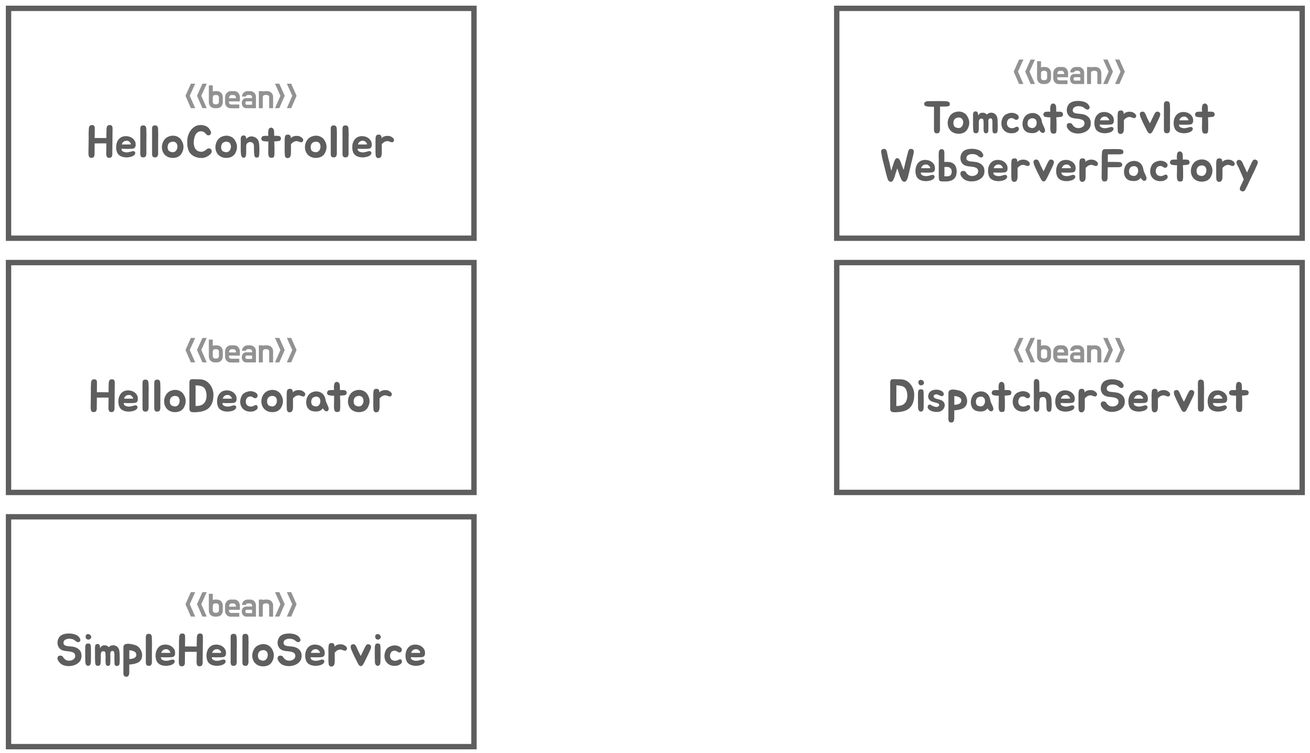
@AutoConfiguration

Meta Annotation
Composed Annotation

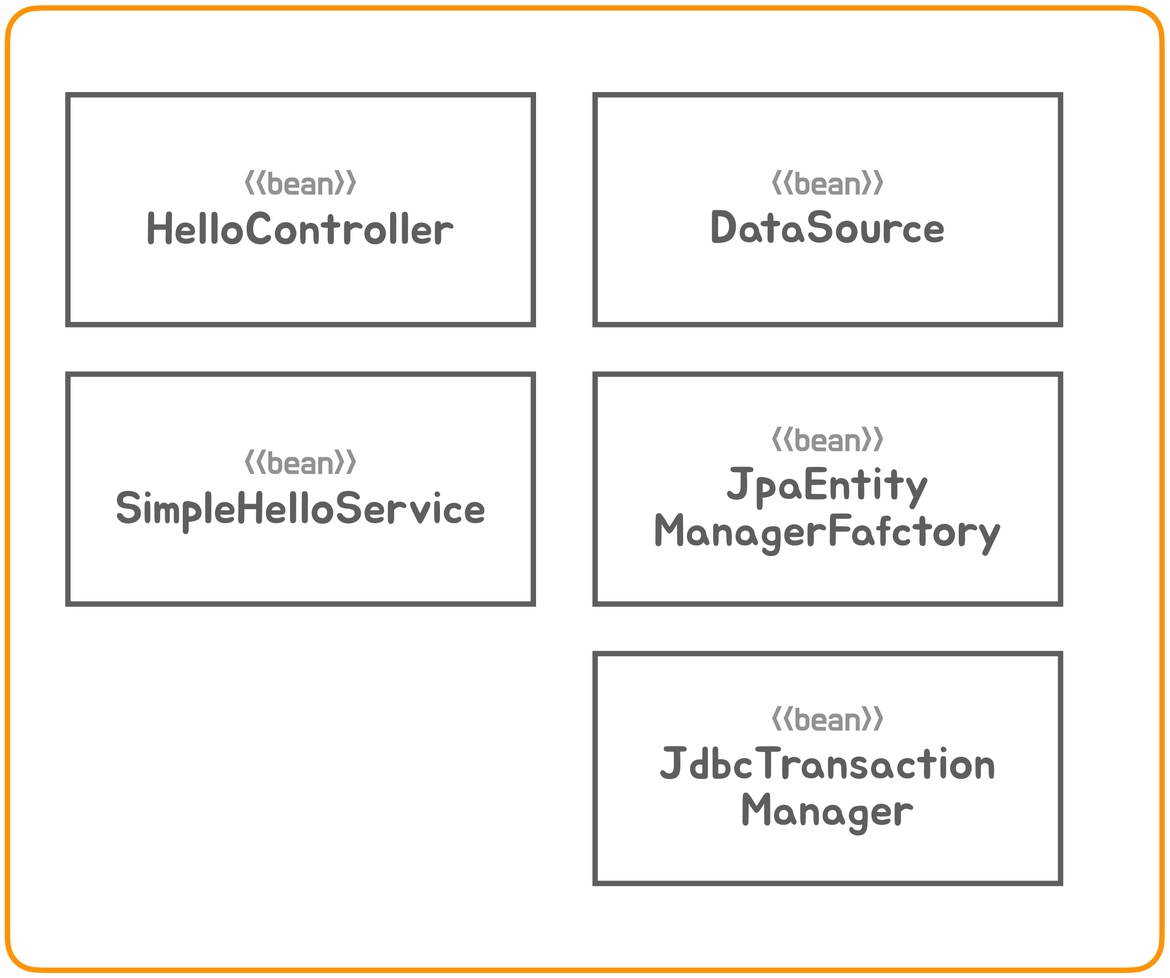

《Annotation》
@Component
《meta》
《meta》
《Annotation》
@Controller
《Annotation》
@Service


《Annotation》
@Component
《meta》
《Annotation》
@Controller
《Annotation》
@ResponseBody
《meta》
《Annotation》
@RestController
《meta》

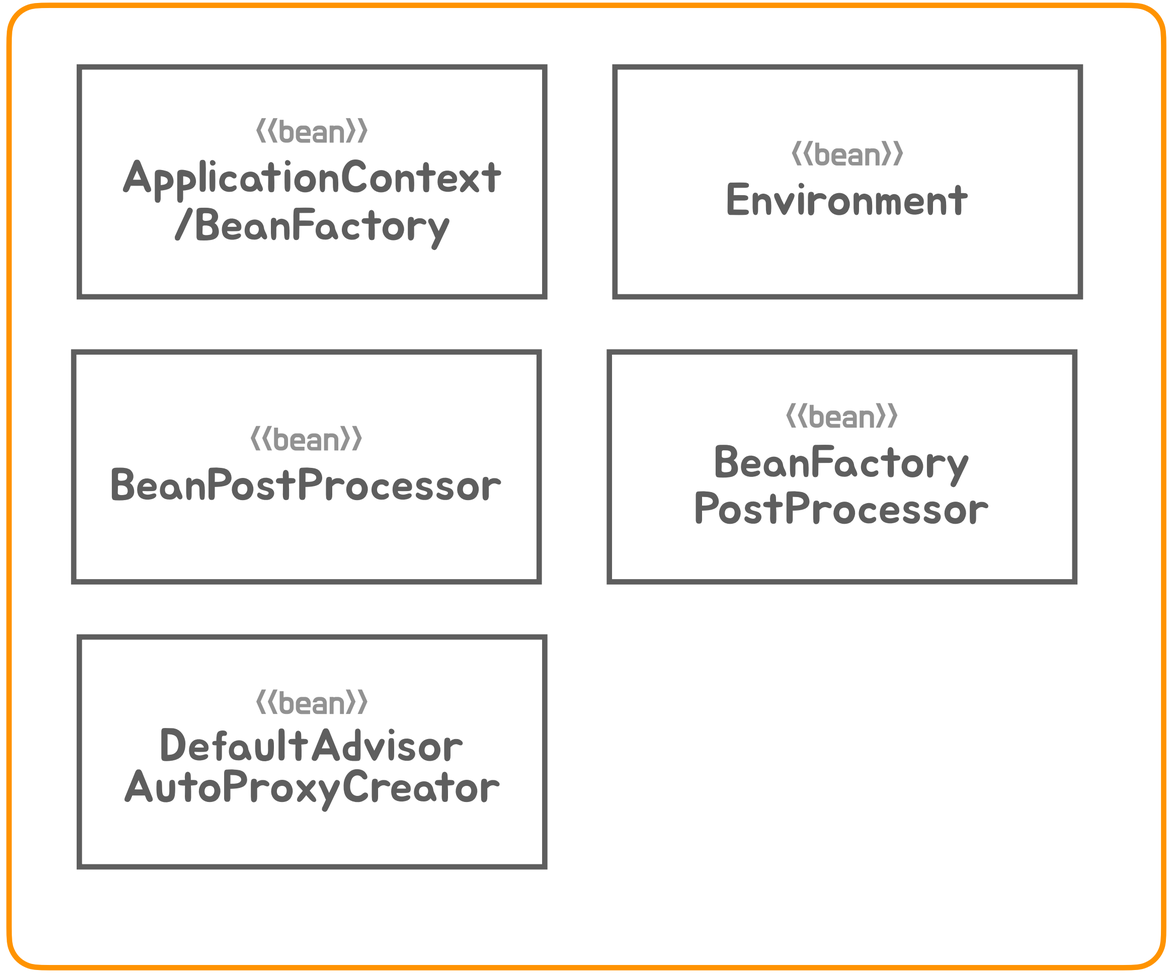
빈 오브젝트와 역할과 구분

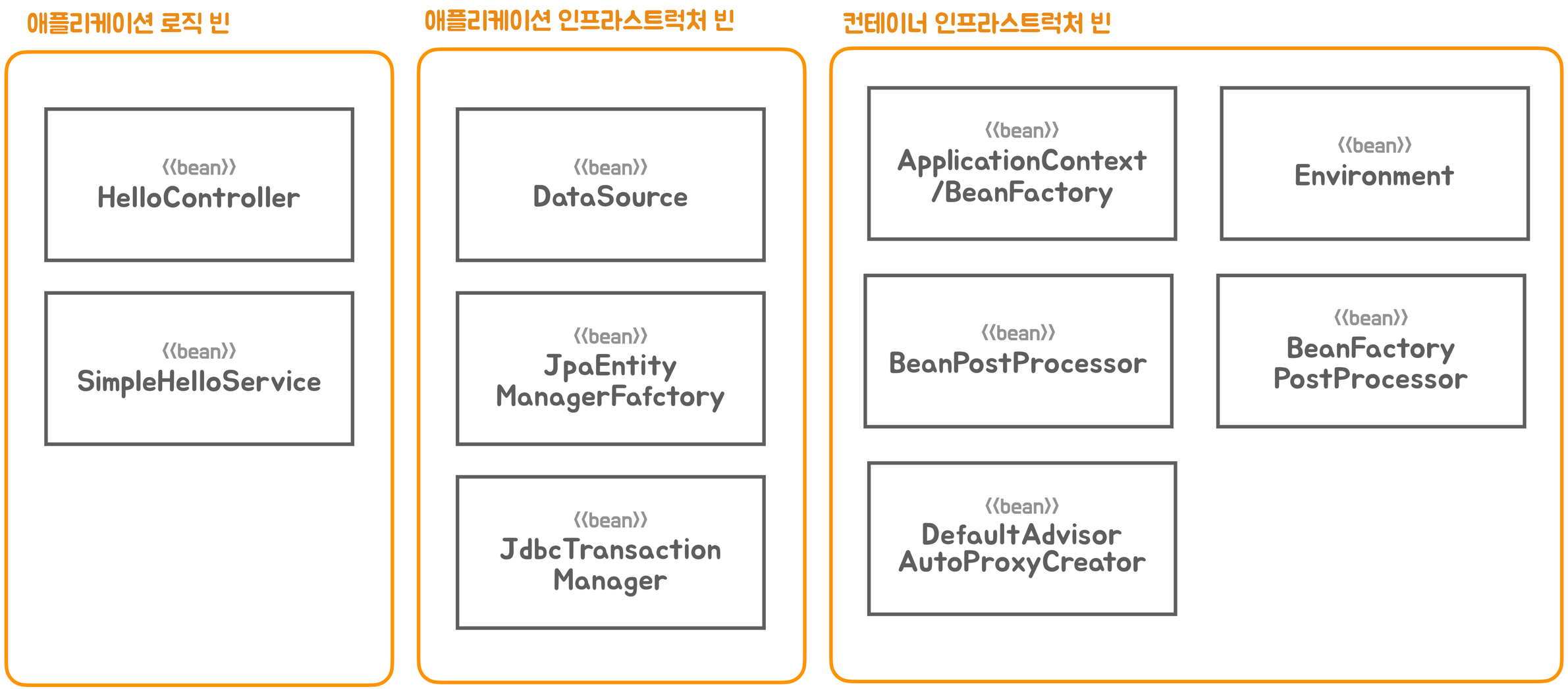

«bean»
HelloController
«bean»
HelloDecorator
«bean»
SimpleHelloService
«bean»
TomcatServlet
WebServerFactory
«bean»
DispatcherServlet

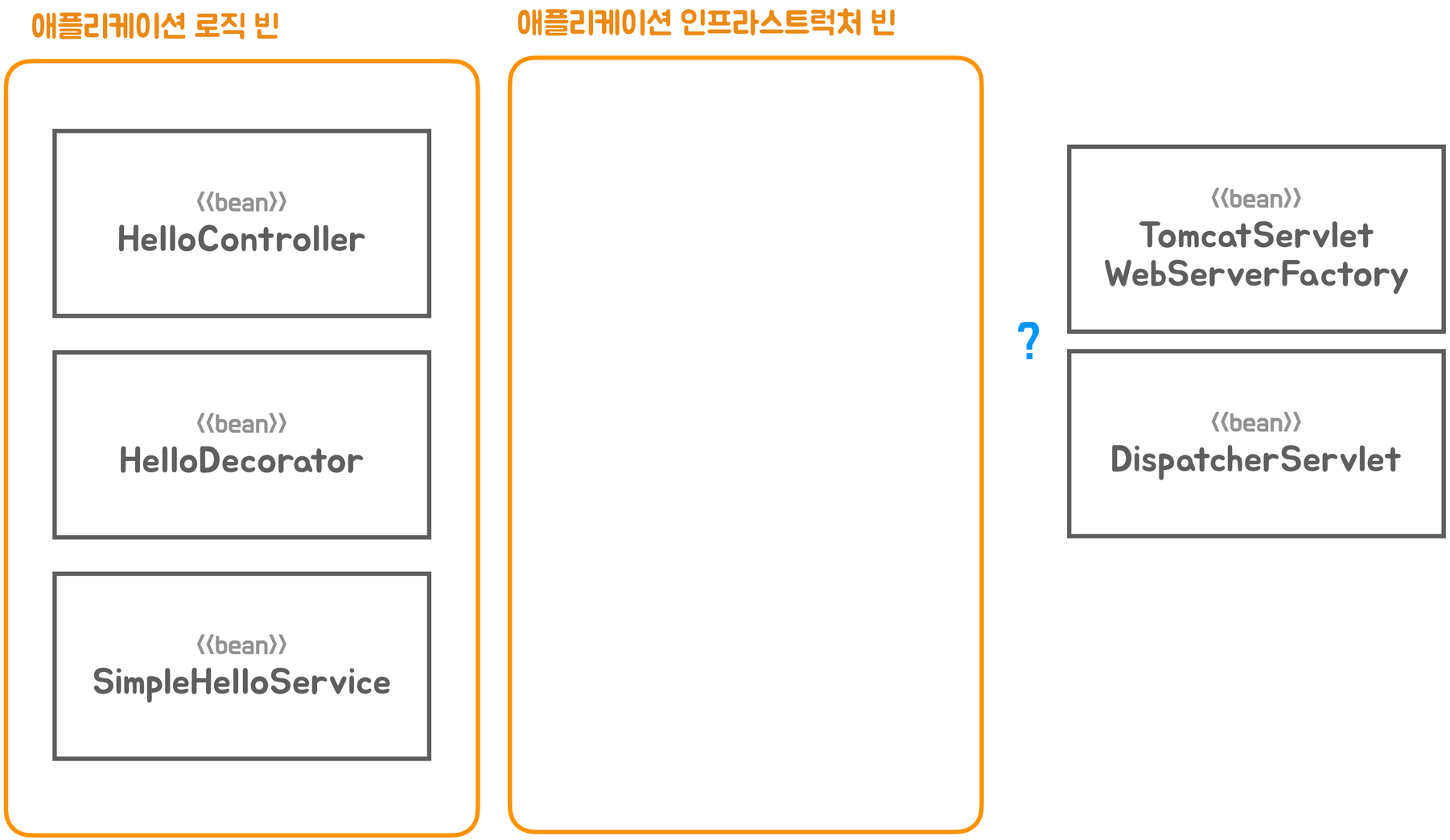
애플리케이션 빈

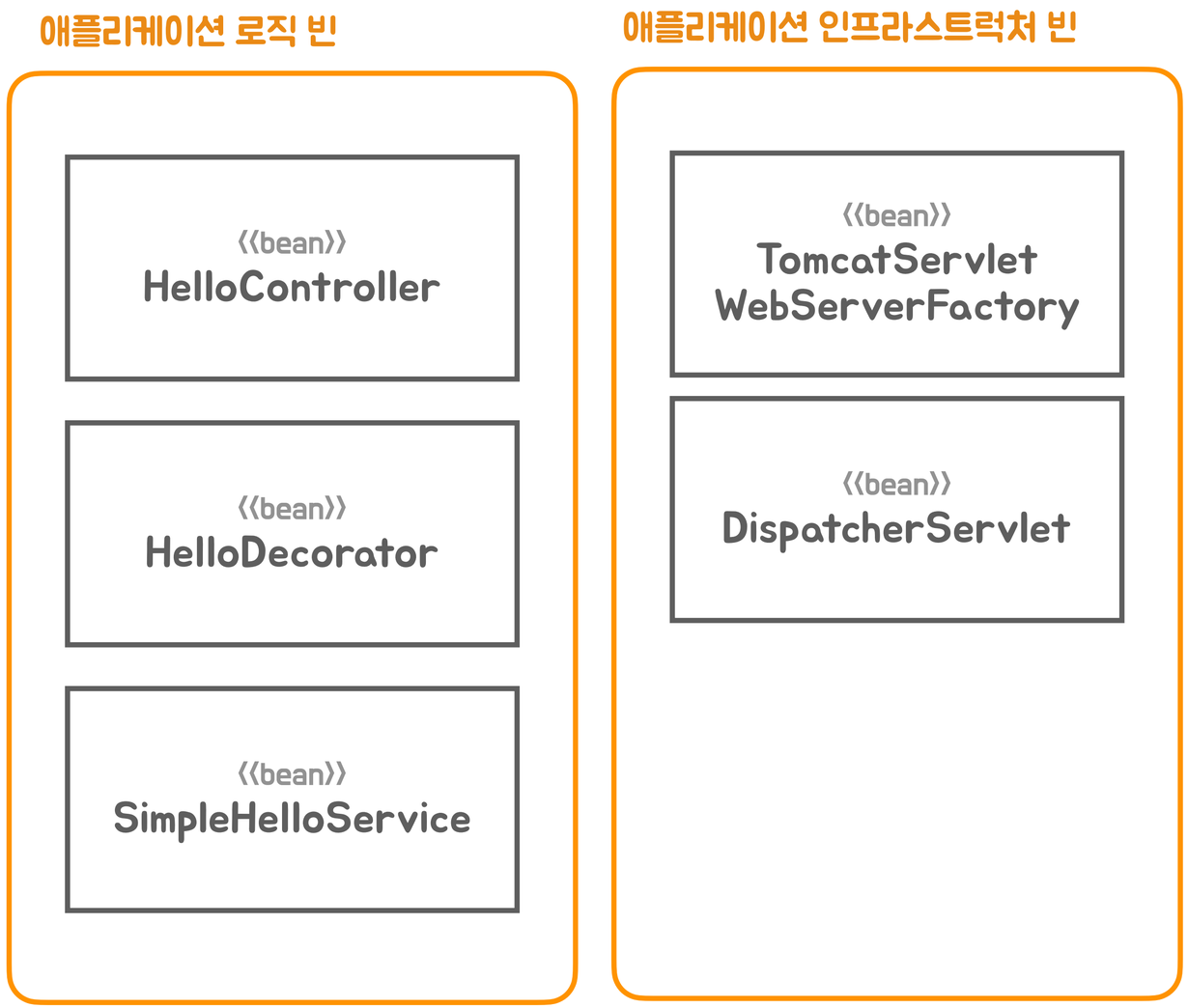


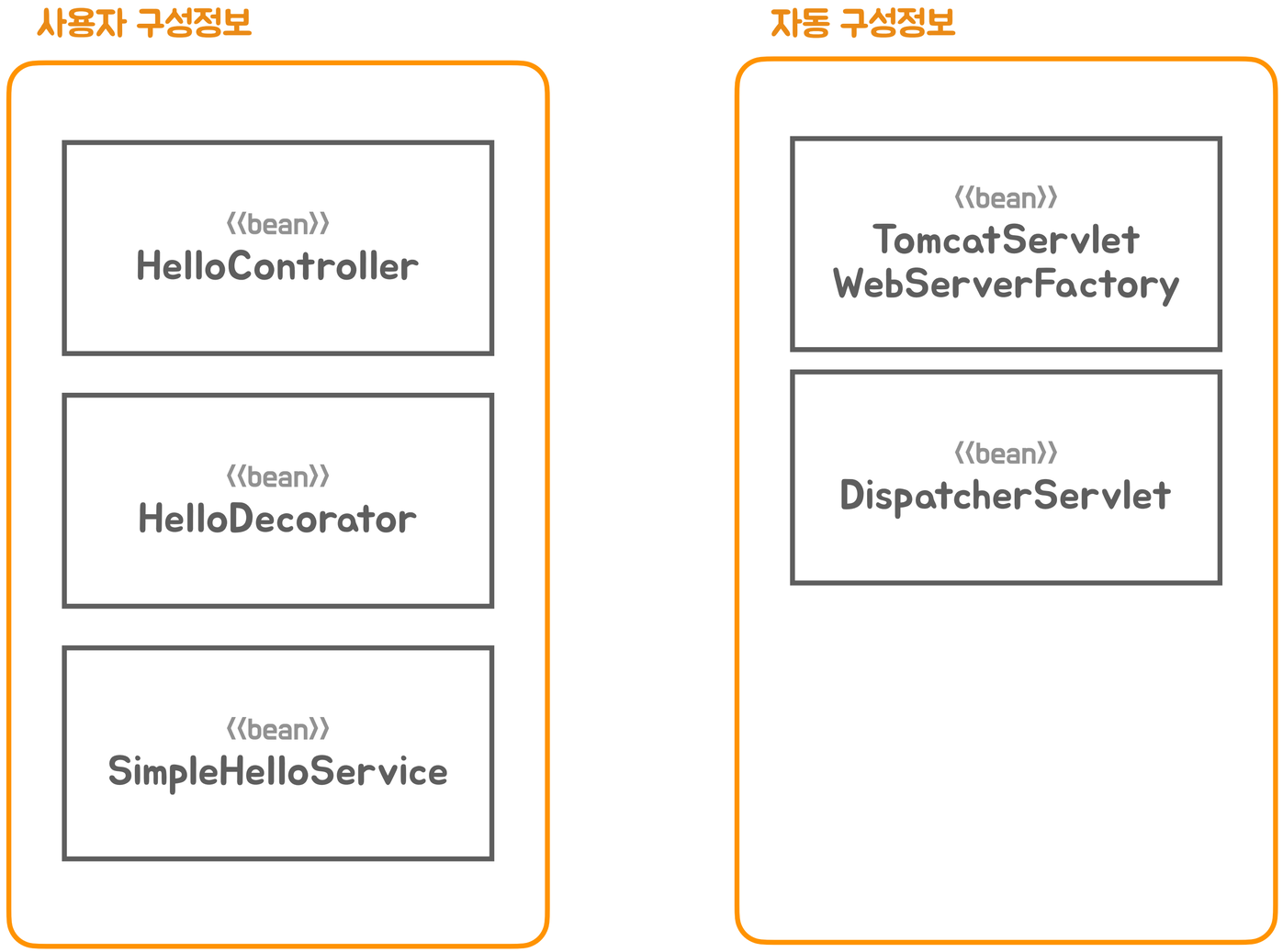
컨테이너 인프라스트럭처 빈

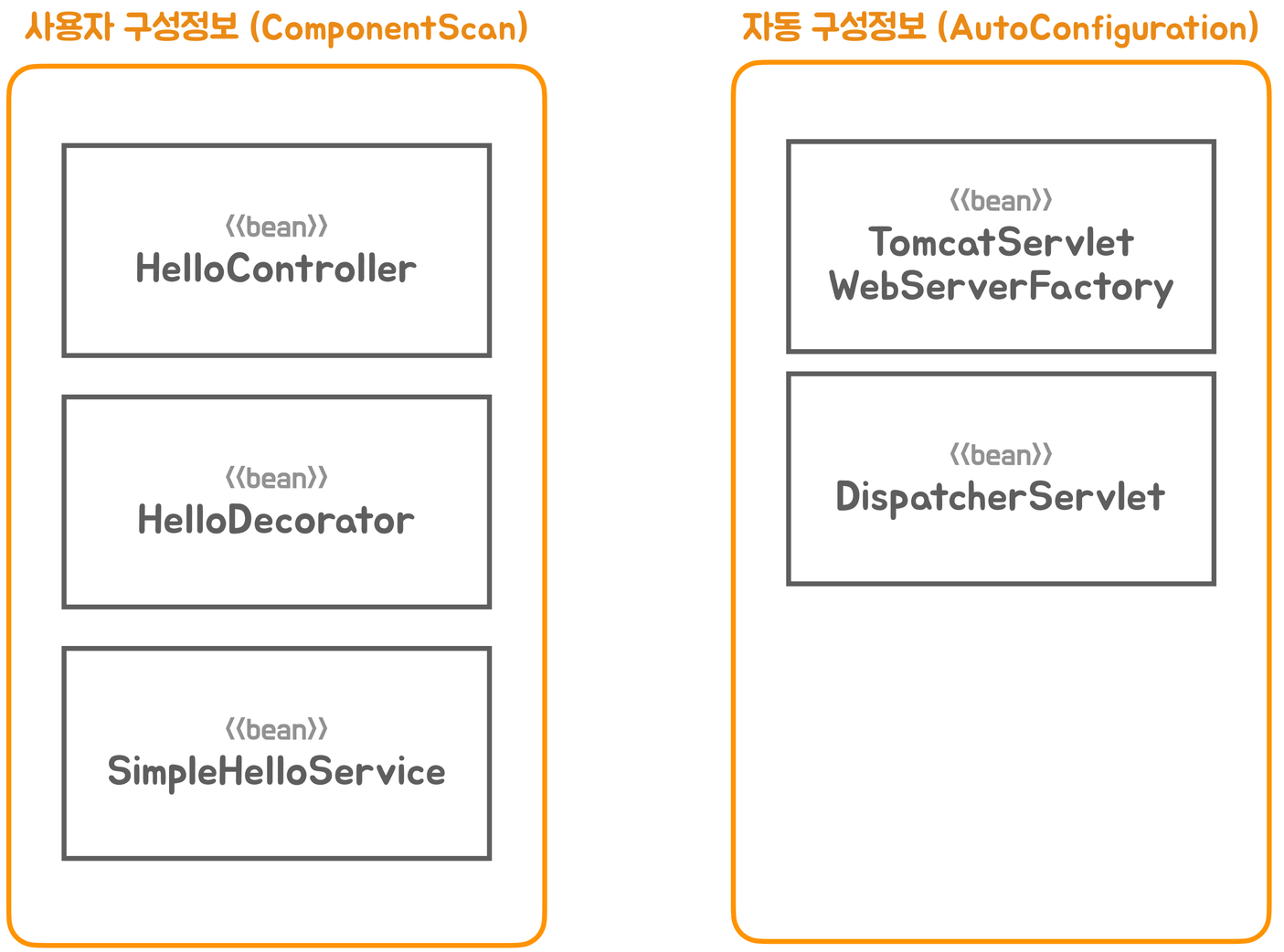


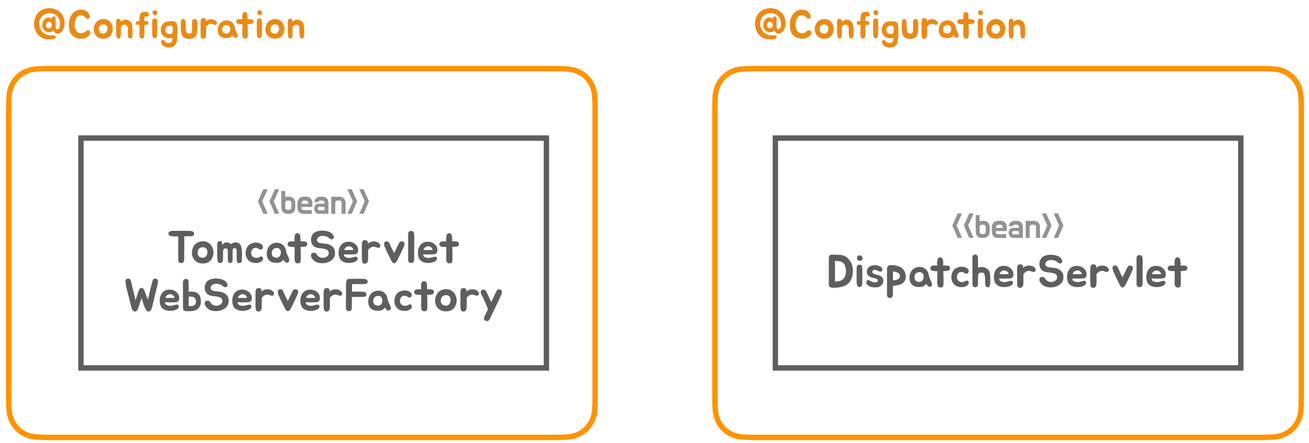

애플리케이션 로직 빈
<<bean>>
HelloController
<<bean>>
SimpleHelloService
애플리케이션 인프라스트럭처 빈
<<bean>>
DataSource
<<bean>>
JpaEntity
ManagerFafctory
<<bean>>
JdbcTransaction
Manager
컨테이너 인프라스트럭처 빈
<<bean>>
ApplicationContext
/BeanFactory
<<bean>>
Environment
<<bean>>
BeanPostProcessor
<<bean>>
BeanFactory
PostProcessor
<<bean>>
DefaultAdvisor
AutoProxyCreator


애플리케이션 로직 빈
애플리케이션 인프라스트럭처 빈
《bean》
HelloController
《bean》
HelloDecorator
《bean》
SimpleHelloService
?
《bean》
TomcatServlet
WebServerFactory
《bean》
DispatcherServlet

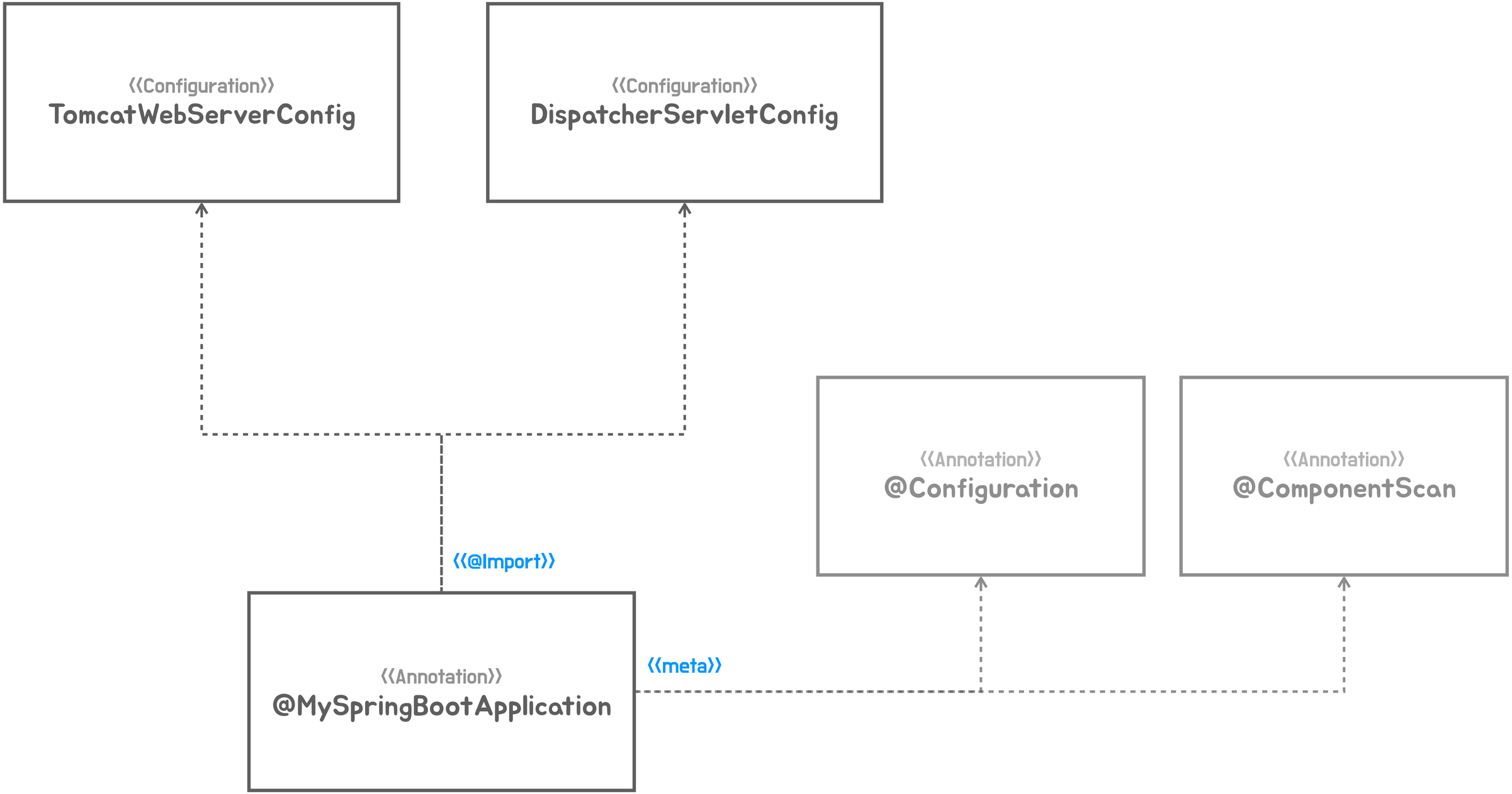

애플리케이션 로직 빈
<<bean>>
HelloController
<<bean>>
HelloDecorator
<<bean>>
SimpleHelloService
애플리케이션 인프라스트럭처 빈
<<bean>>
TomcatServlet
WebServerFactory
<<bean>>
DispatcherServlet

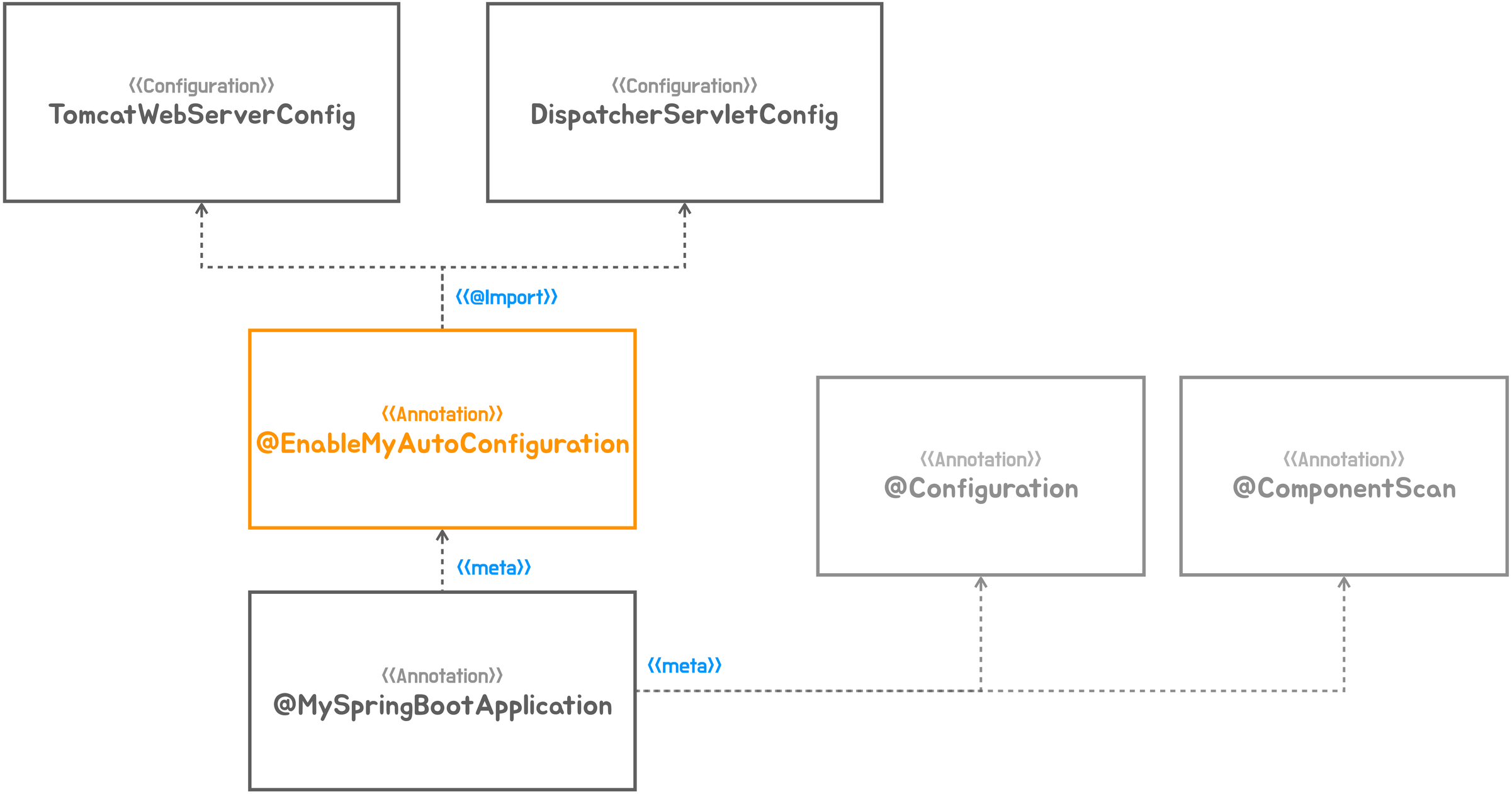

사용자 구성정보
<<bean>>
HelloController
<<bean>>
HelloDecorator
<<bean>>
SimpleHelloService
자동 구성정보
<<bean>>
TomcatServlet
WebServerFactory
<<bean>>
DispatcherServlet


사용자 구성정보 (ComponentScan)
<<bean>>
HelloController
<<bean>>
HelloDecorator
<<bean>>
SimpleHelloService
자동 구성정보 (AutoConfiguration)
<<bean>>
TomcatServlet
WebServerFactory
<<bean>>
DispatcherServlet

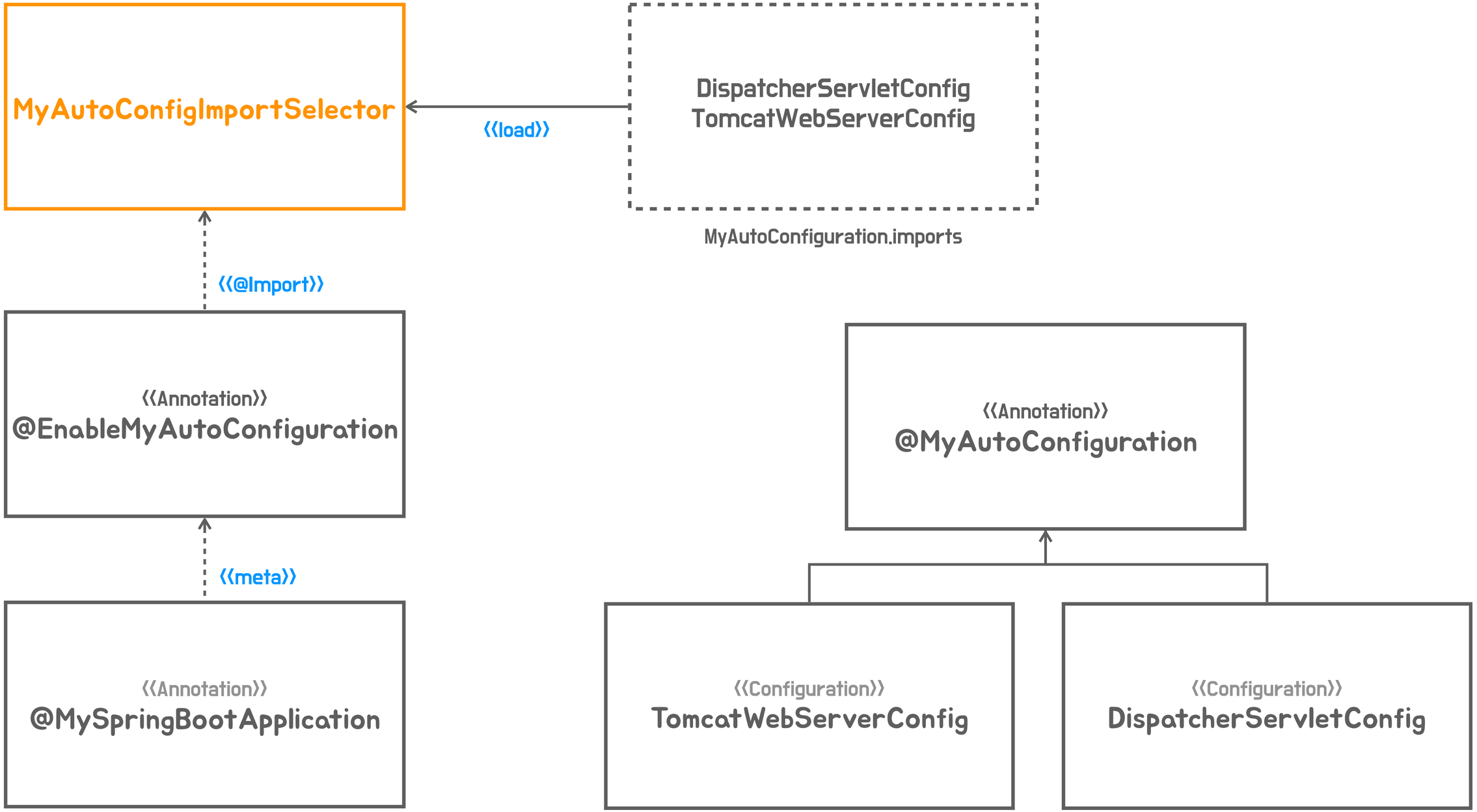

@Configuration
《bean》
TomcatServlet
WebServerFactory
@Configuration
《bean》
DispatcherServlet

인프라 빈 구성 정보의 분리


《Configuration》
TomcatWebServerConfig
《Configuration》
DispatcherServletConfig
《Annotation》
@Configuration
《Annotation》
@ComponentScan
《@Import》
《Annotation》
@MySpringBootApplication
《meta》


《Configuration》
TomcatWebServerConfig
《Configuration》
DispatcherServletConfig
《@Import》
《Annotation》
@EnableMyAutoConfiguration
《Annotation》
@Configuration
《Annotation》
@ComponentScan
《meta》
《Annotation》
@MySpringBootApplication
《meta》

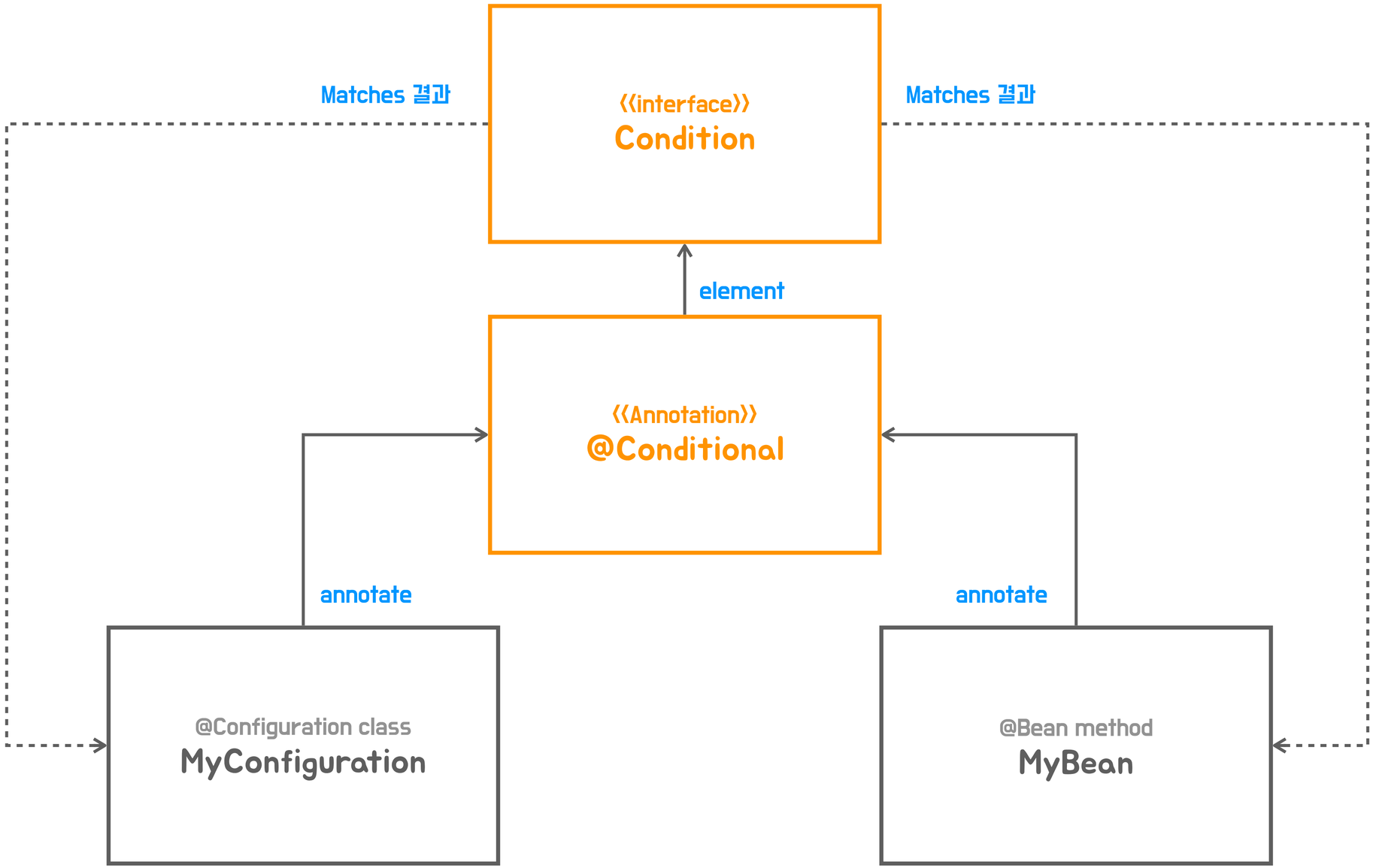
동적인 자동 구성 정보 등록

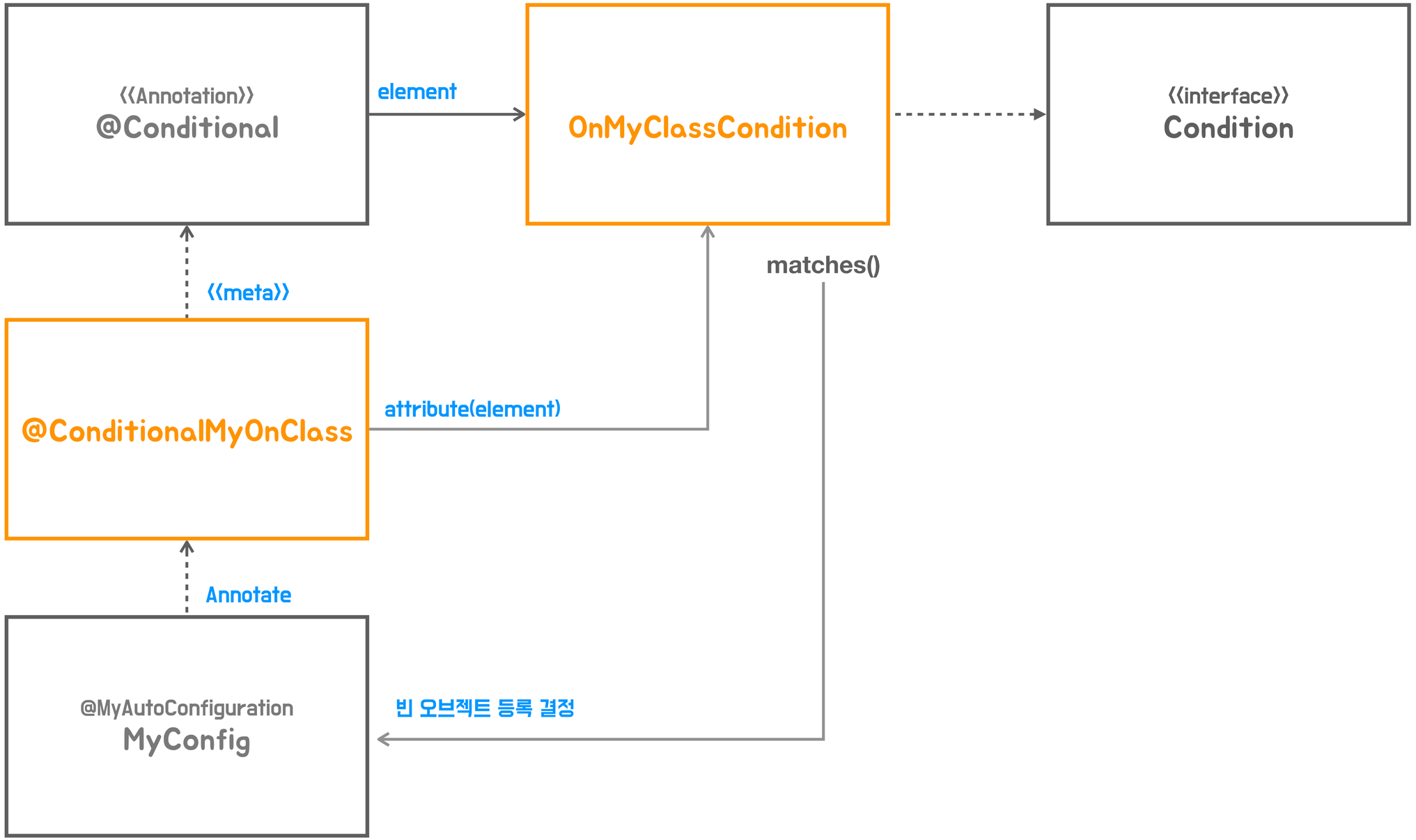

MyAutoConfigImportSelector
DispatcherServletConfig
TomcatWebServerConfig
MyAutoConfiguration.imports
<<load>>
<<@Import>>
<<Annotation>>
@EnableMyAutoConfiguration
<<meta>>
<<Annotation>>
@MySpringBootApplication
<<Annotation>>
@MyAutoConfiguration
<<Configuration>>
TomcatWebServerConfig
<<Configuration>>
DispatcherServletConfig

조건부 자동 구성

@Conditional

@Conditional과 Condition

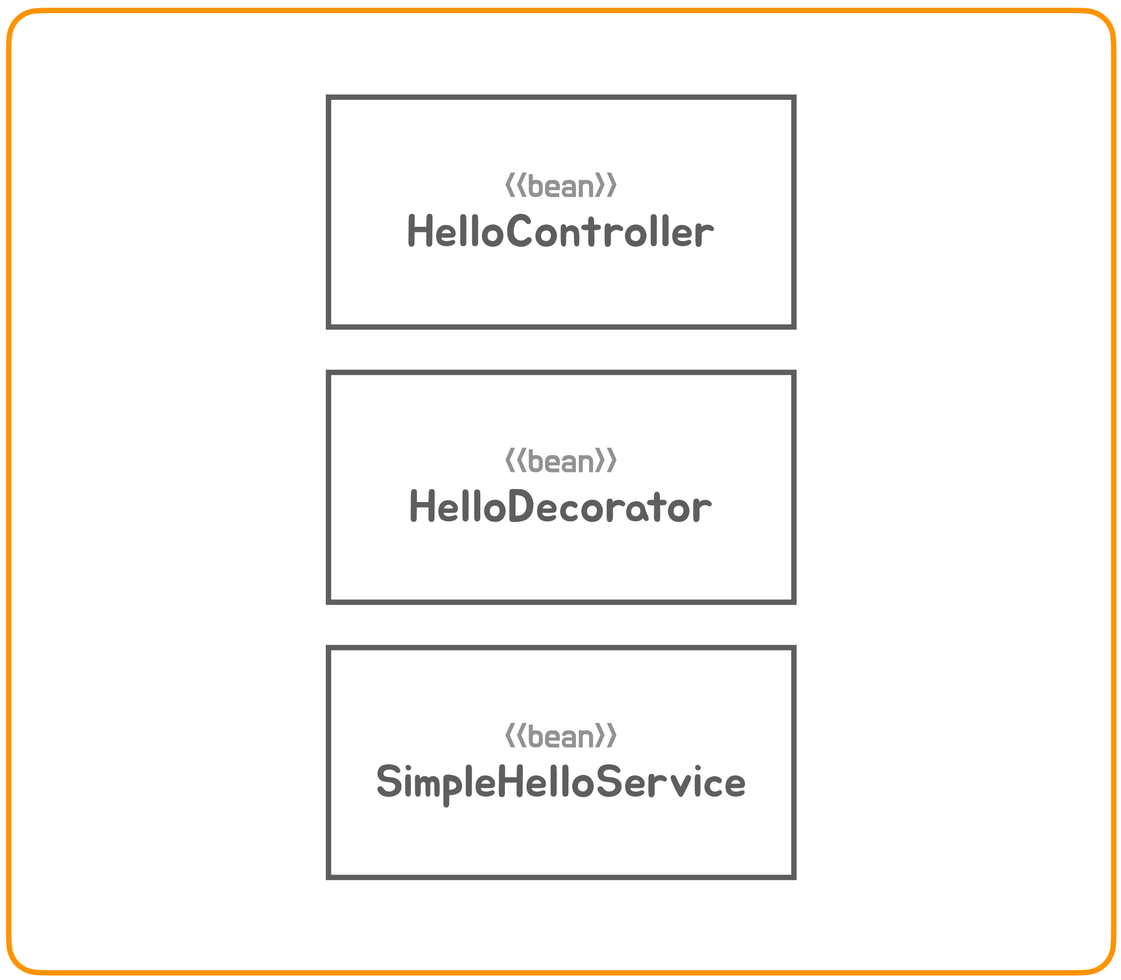

Matches 결과
<<interface>>
Condition
Matches 결과
element
<<Annotation>>
@Conditional
annotate
annotate
@Configuration class
MyConfiguration
@Bean method
MyBean


《Annotation》
@Conditional
element
OnMyClassCondition
《interface》
Condition
《meta》
matches()
@ConditionalMyOnClass
attribute(element)
Annotate
@MyAutoConfiguration
MyConfig
빈 오브젝트 등록 결정

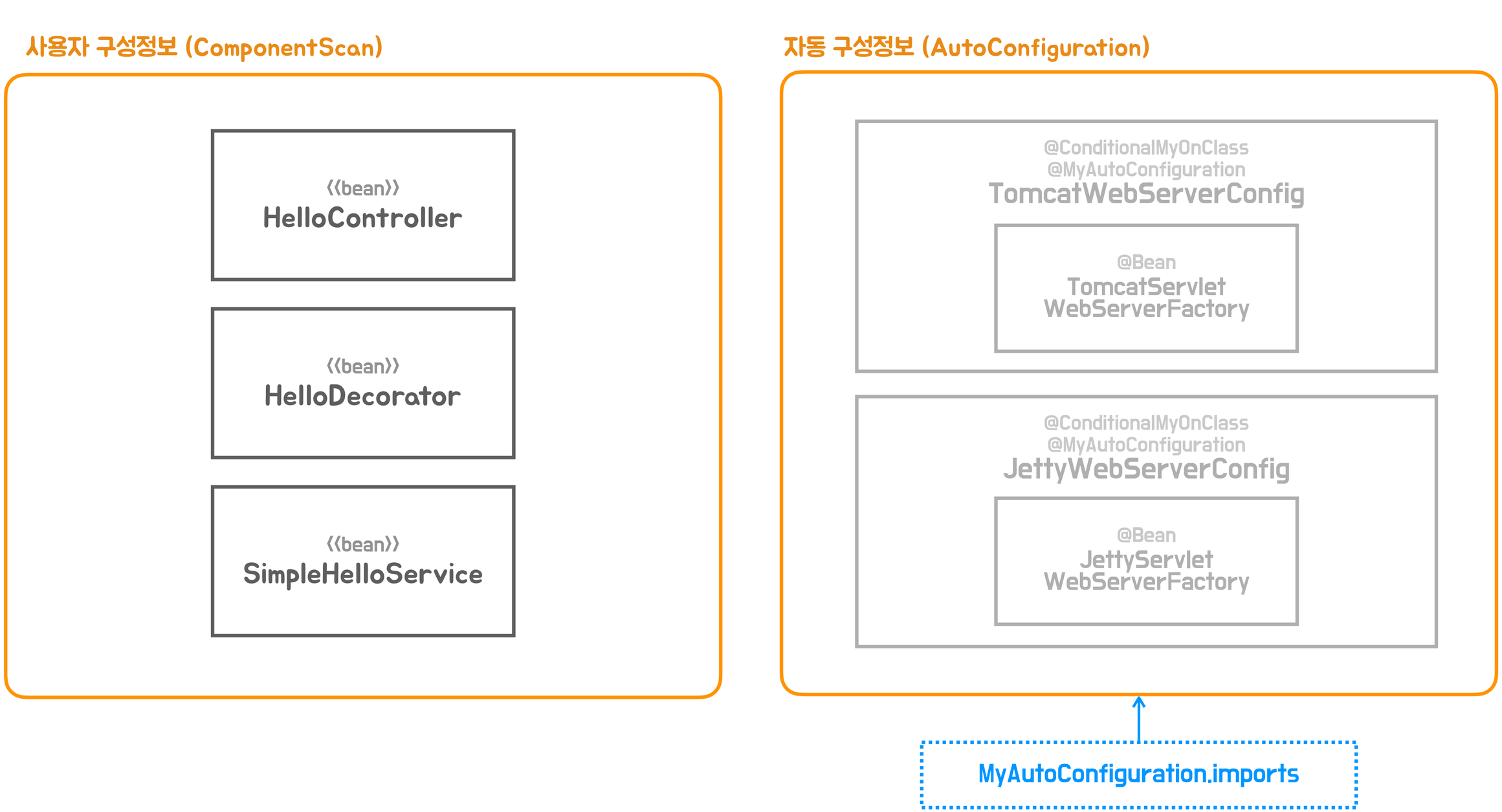
커스톰 빈 구성 정보 활용

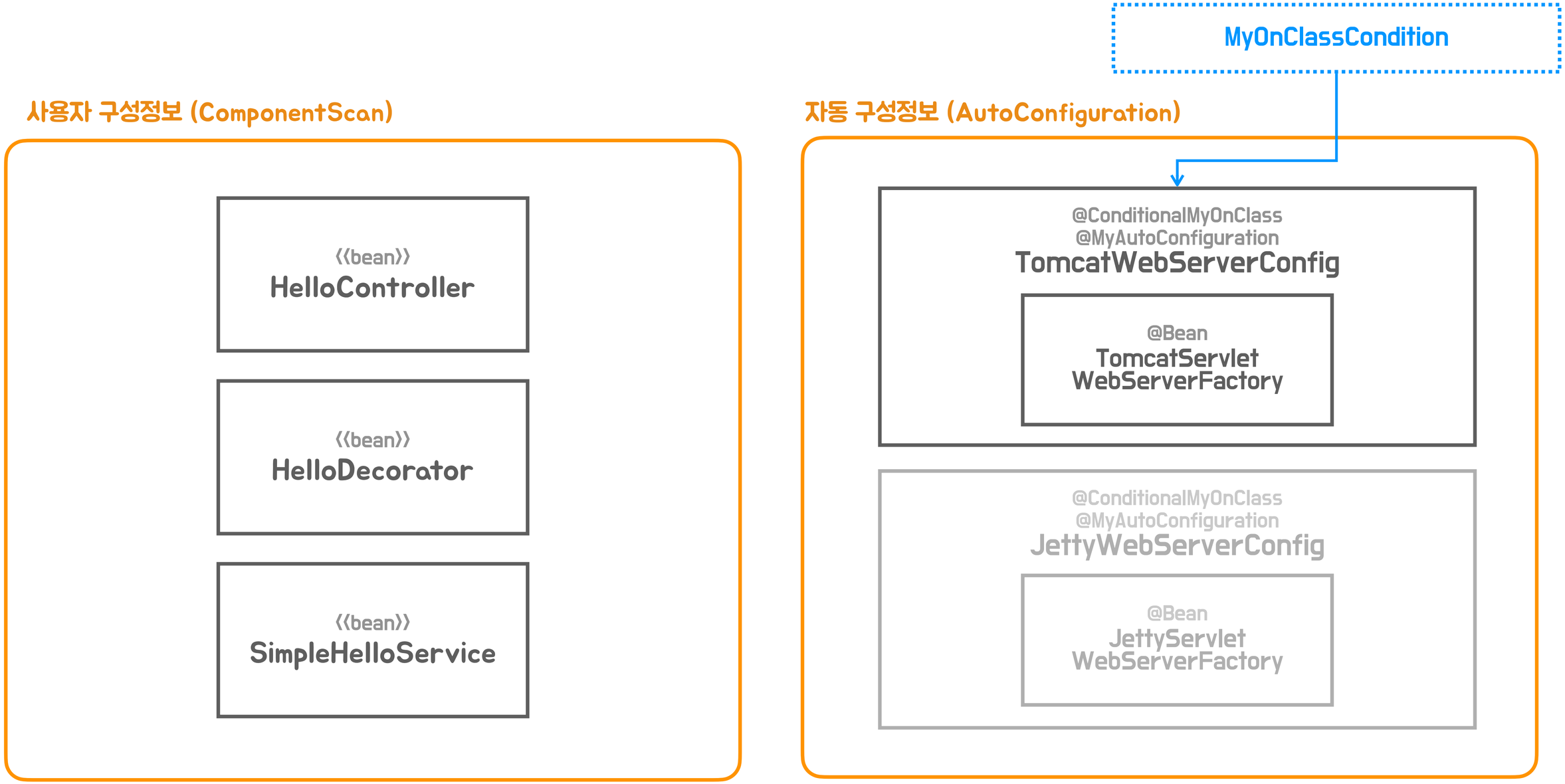
@ConditionalOnMissingBean

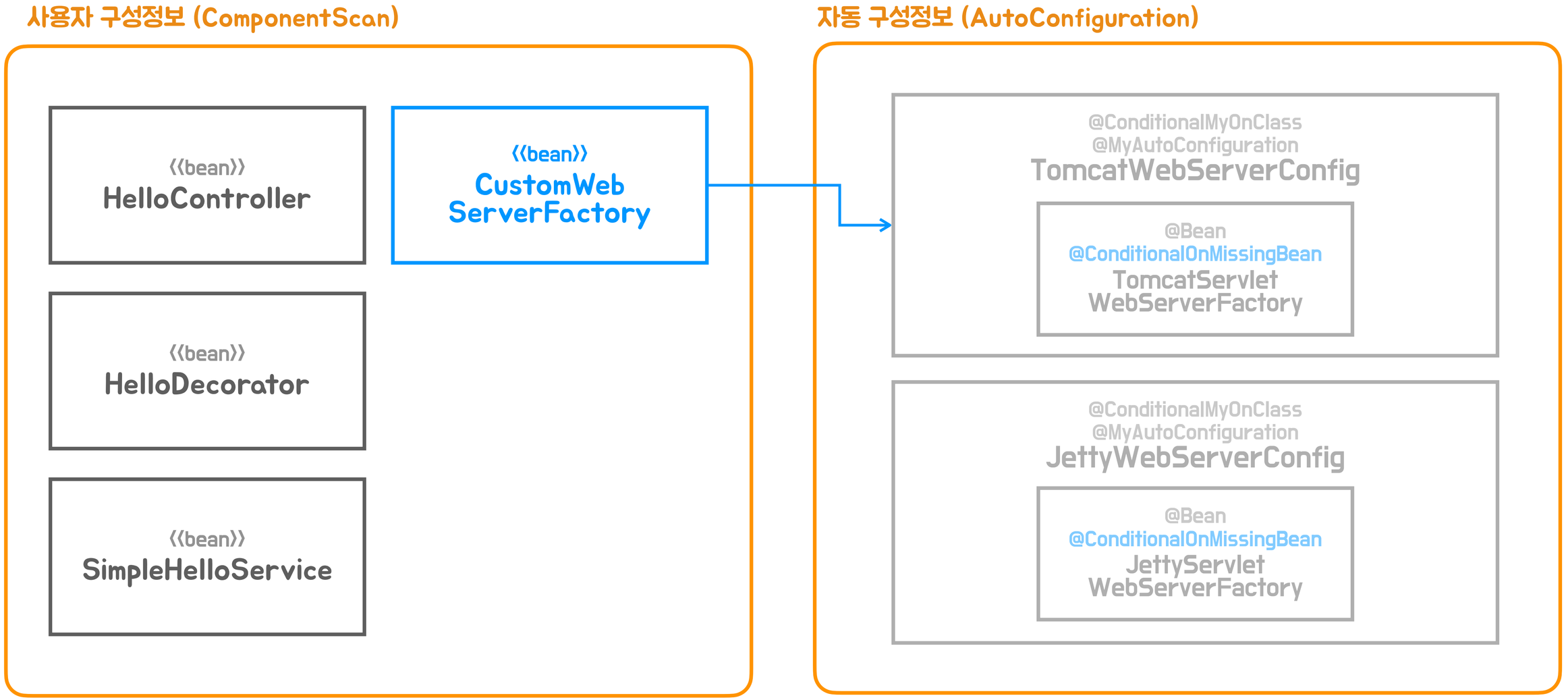
사용자 구성정보 (ComponentScan)

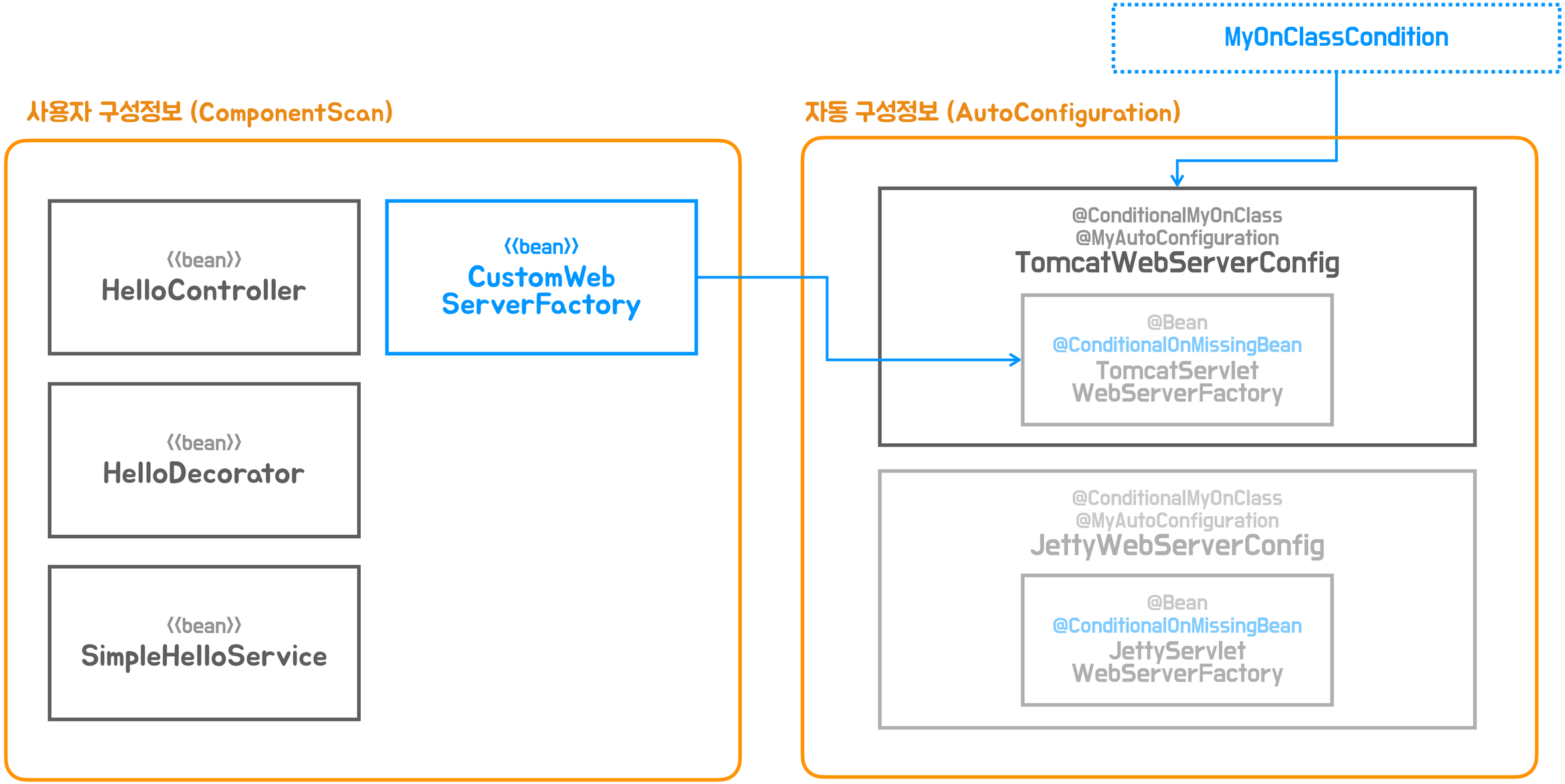


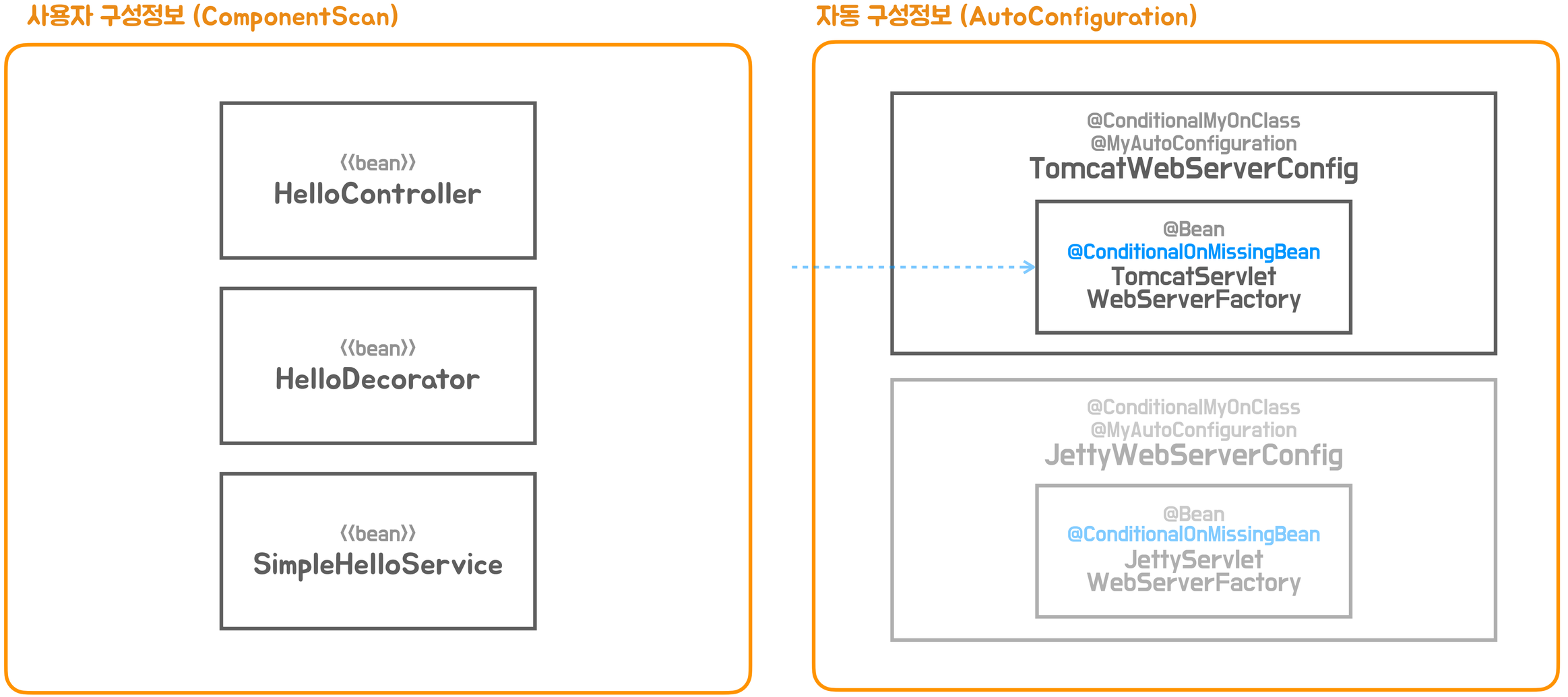
자동 구성정보 (AutoConfiguration)

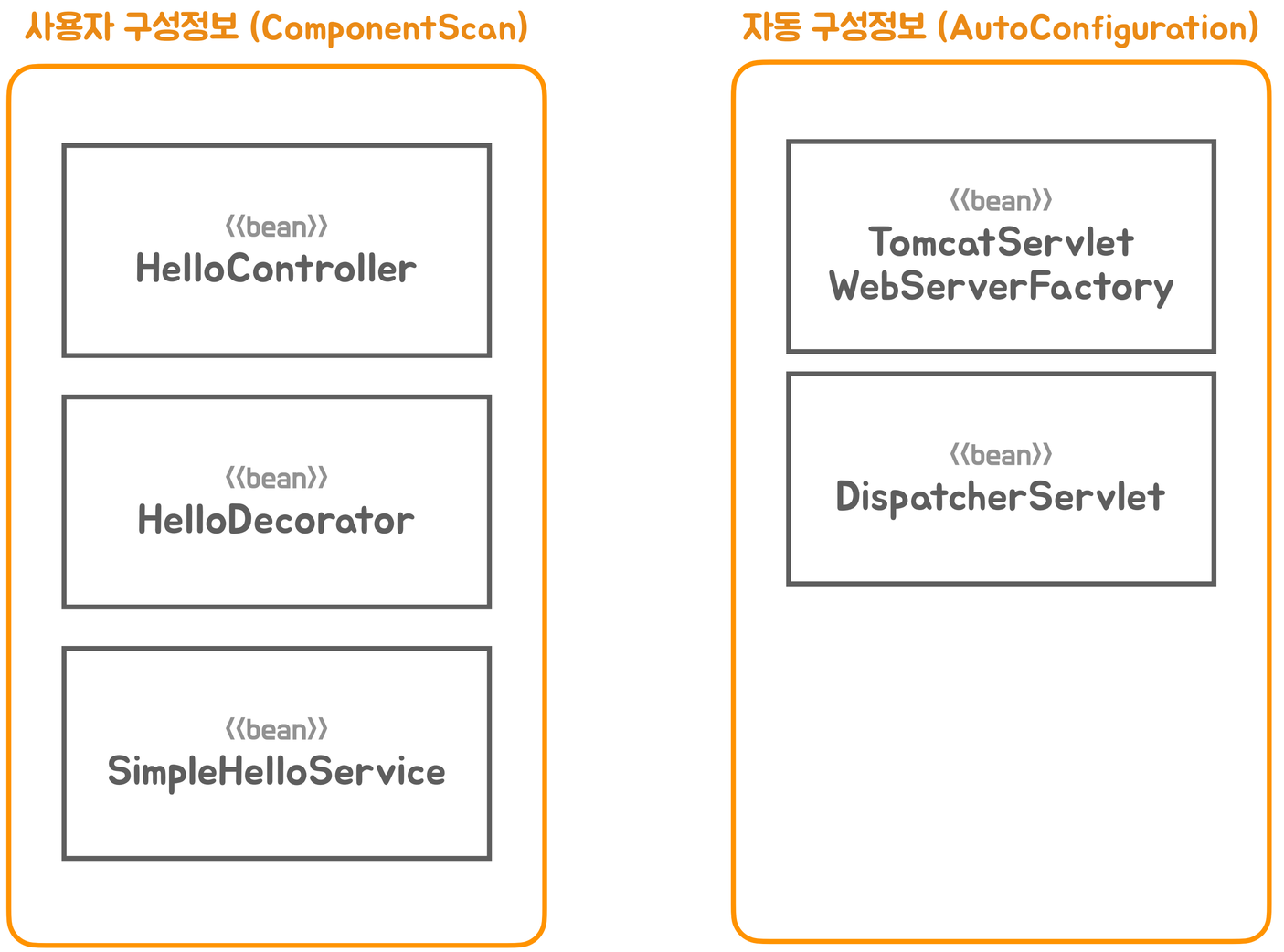

사용자 구성정보 (ComponentScan)
<<bean>>
HelloController
<<bean>>
HelloDecorator
<<bean>>
SimpleHelloService
자동 구성정보 (AutoConfiguration)
@ConditionalMyOnClass
@MyAutoConfiguration
TomcatWebServerConfig
@Bean
TomcatServlet
WebServerFactory
@ConditionalMyOnClass
@MyAutoConfiguration
JettyWebServerConfig
@Bean
JettyServlet
WebServerFactory
MyAutoConfiguration.imports


MyOnClassCondition
사용자 구성정보 (ComponentScan)
<<bean>>
HelloController
<<bean>>
HelloDecorator
<<bean>>
SimpleHelloService
자동 구성정보 (AutoConfiguration)
@ConditionalMyOnClass
@MyAutoConfiguration
TomcatWebServerConfig
@Bean
TomcatServlet
WebServerFactory
@ConditionalMyOnClass
@MyAutoConfiguration
JettyWebServerConfig
@Bean
JettyServlet
WebServerFactory


사용자 구성정보 (ComponentScan)
자동 구성정보 (AutoConfiguration)
《bean》
HelloController
《bean》
CustomWeb
ServerFactory
《bean》
HelloDecorator
《bean》
SimpleHelloService
@ConditionalMyOnClass
@MyAutoConfiguration
TomcatWebServerConfig
@Bean
@ConditionalOnMissingBean
TomcatServlet
WebServerFactory
@ConditionalMyOnClass
@MyAutoConfiguration
JettyWebServerConfig
@Bean
@ConditionalOnMissingBean
JettyServlet
WebServerFactory

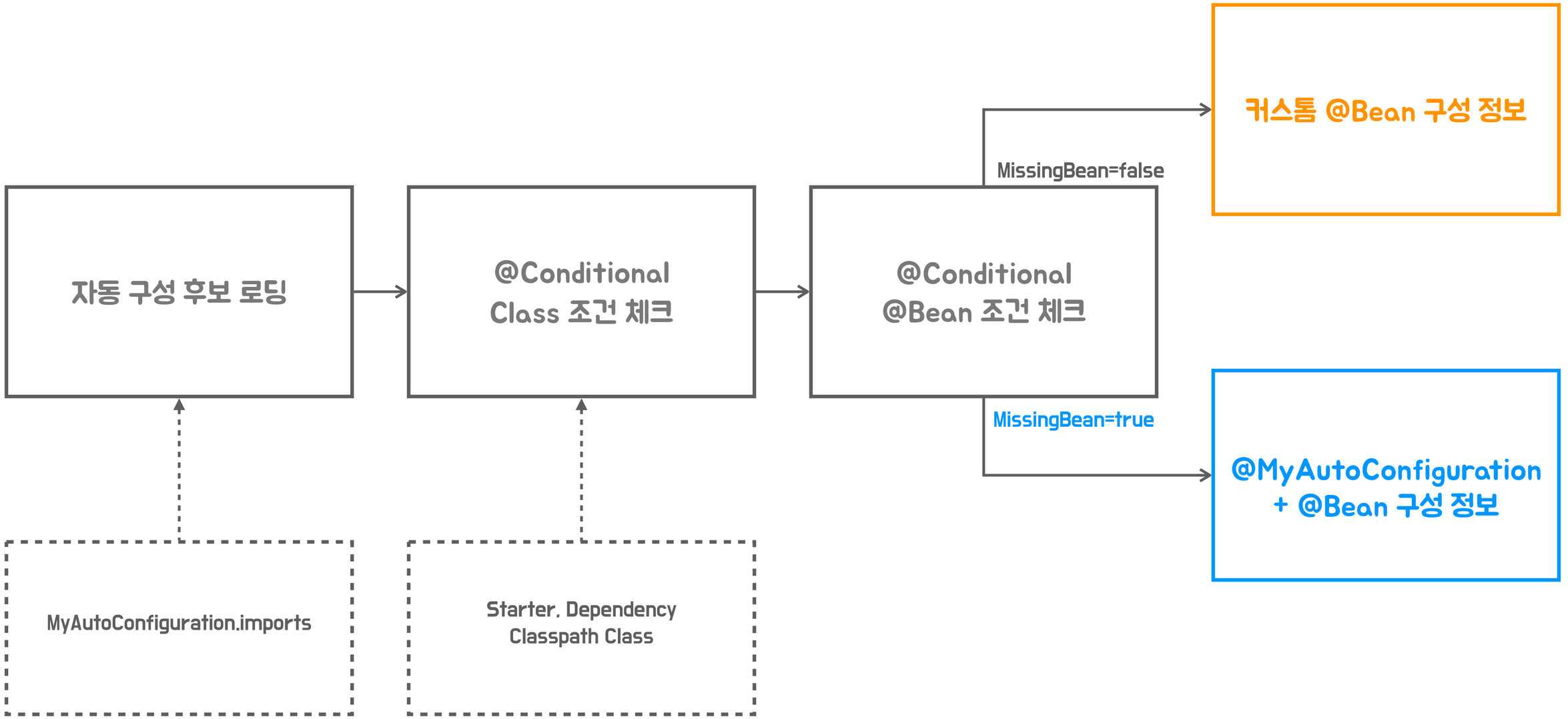

MyOnClassCondition
사용자 구성정보 (ComponentScan)
자동 구성정보 (AutoConfiguration)
《bean》
HelloController
《bean》
CustomWeb
ServerFactory
《bean》
HelloDecorator
《bean》
SimpleHelloService
@ConditionalMyOnClass
@MyAutoConfiguration
TomcatWebServerConfig
@Bean
@ConditionalOnMissingBean
TomcatServlet
WebServerFactory
@ConditionalMyOnClass
@MyAutoConfiguration
JettyWebServerConfig
@Bean
@ConditionalOnMissingBean
JettyServlet
WebServerFactory

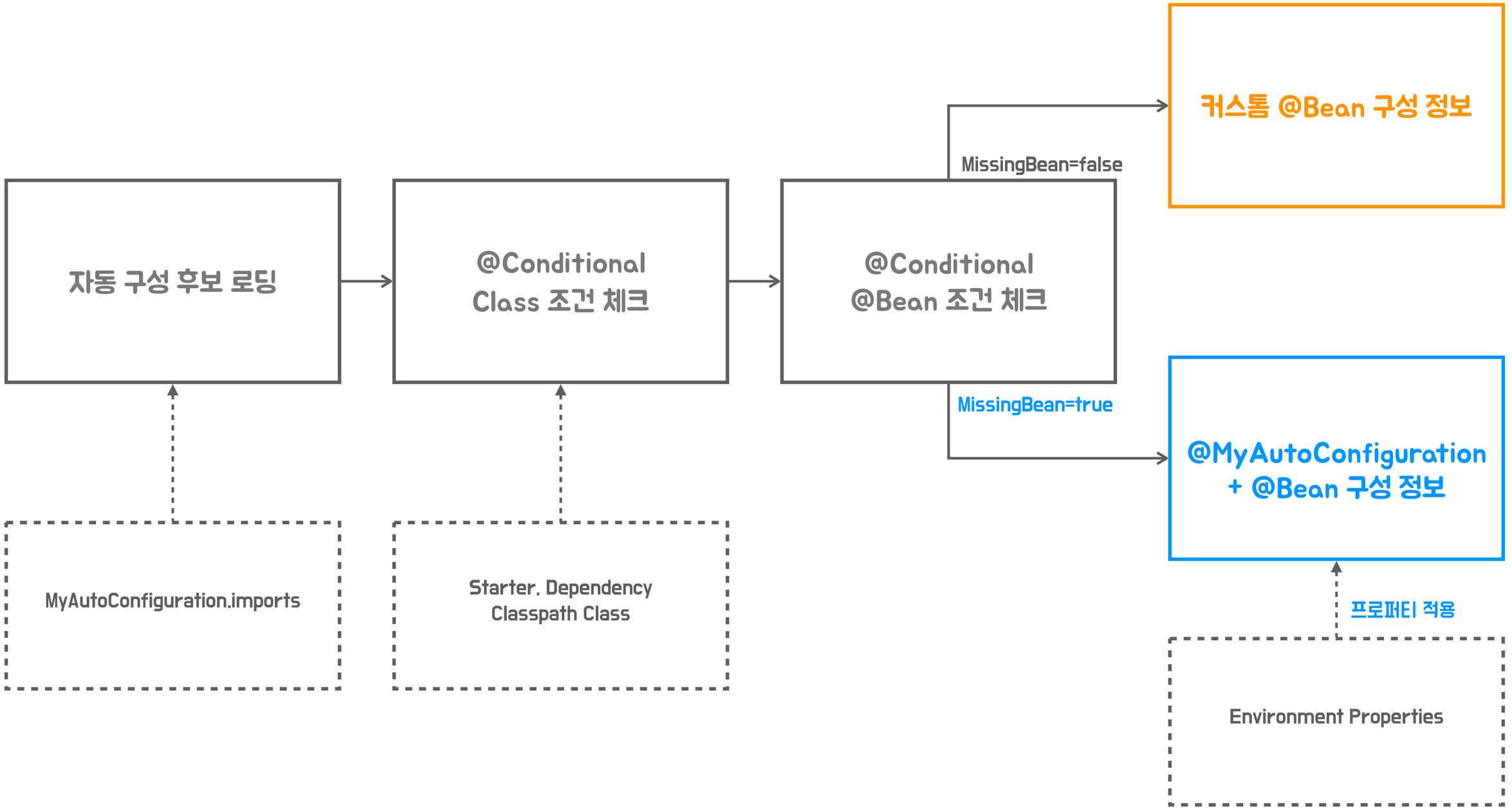

사용자 구성정보 (ComponentScan)
《bean》
HelloController
《bean》
HelloDecorator
《bean》
SimpleHelloService
자동 구성정보 (AutoConfiguration)
@ConditionalMyOnClass
@MyAutoConfiguration
TomcatWebServerConfig
@Bean
@ConditionalOnMissingBean
TomcatServlet
WebServerFactory
@ConditionalMyOnClass
@MyAutoConfiguration
JettyWebServerConfig
@Bean
@ConditionalOnMissingBean
JettyServlet
WebServerFactory


사용자 구성정보 (ComponentScan)
<<bean>>
HelloController
<<bean>>
HelloDecorator
<<bean>>
SimpleHelloService
자동 구성정보 (AutoConfiguration)
<<bean>>
TomcatServlet
WebServerFactory
<<bean>>
DispatcherServlet

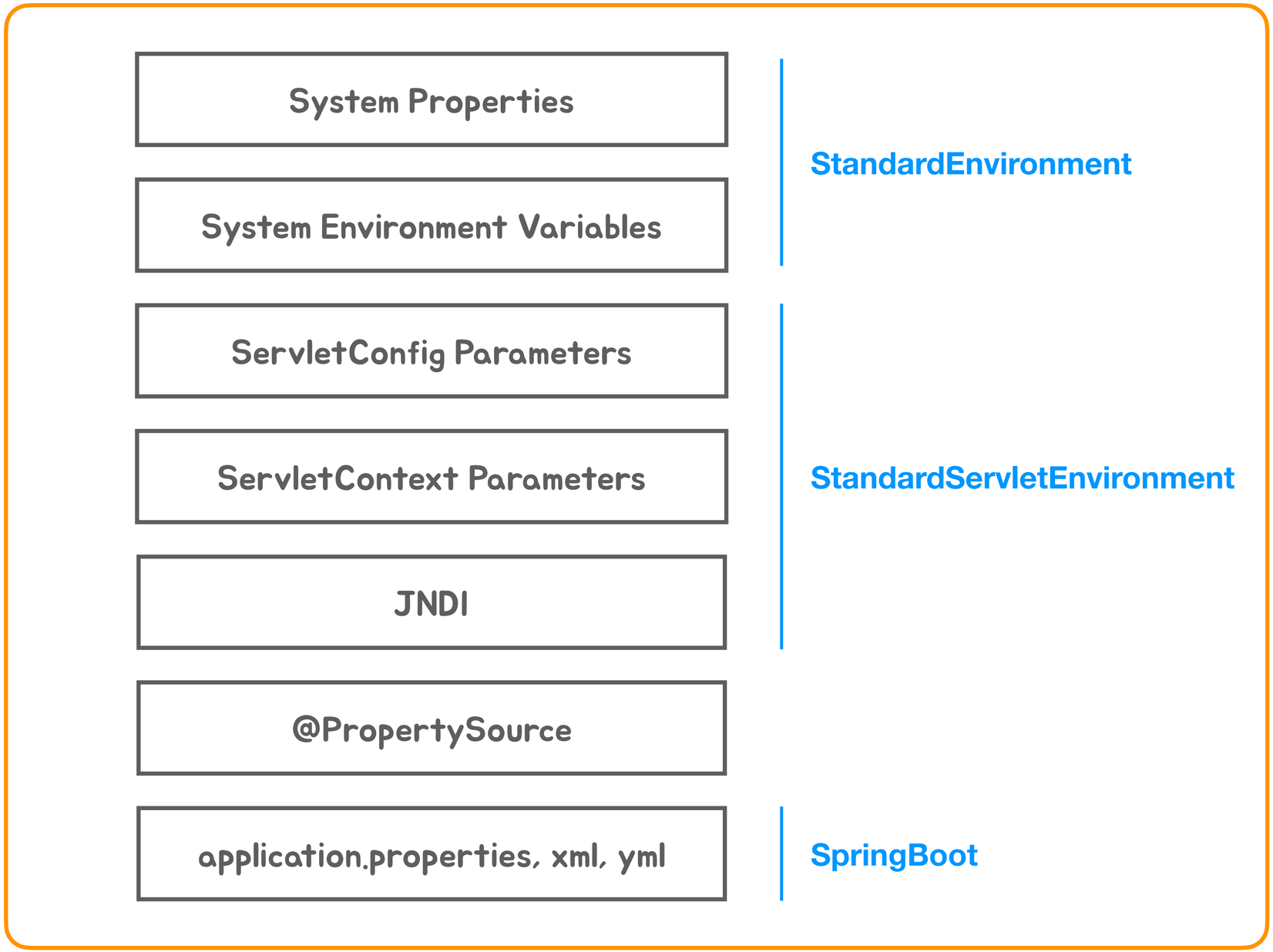
외부 설정을 활용하는 자동 구성

Environment Abstraction


자동 구성 후보 로딩
@Conditional
Class 조건 체크
@Conditional
@Bean 조건 체크
MissingBean=false
커스톰 @Bean 구성 정보
MissingBean=true
@MyAutoConfiguration
+ @Bean 구성 정보
MyAutoConfiguration.imports
Starter, Dependency
Classpath Class


자동 구성 후보 로딩
@Conditional
Class 조건 체크
@Conditional
@Bean 조건 체크
MissingBean=false
커스톰 @Bean 구성 정보
MissingBean=true
@MyAutoConfiguration
+ @Bean 구성 정보
MyAutoConfiguration.imports
Starter, Dependency
Classpath Class
프로퍼티 적용
Environment Properties

Environment Abstraction - Properties



Environment
.getProperty("property.name")

- property.name
- property_name
- PROPERTY.NAME
- PROPERTY_NAME

Spring JDBC 자동 구성 개발

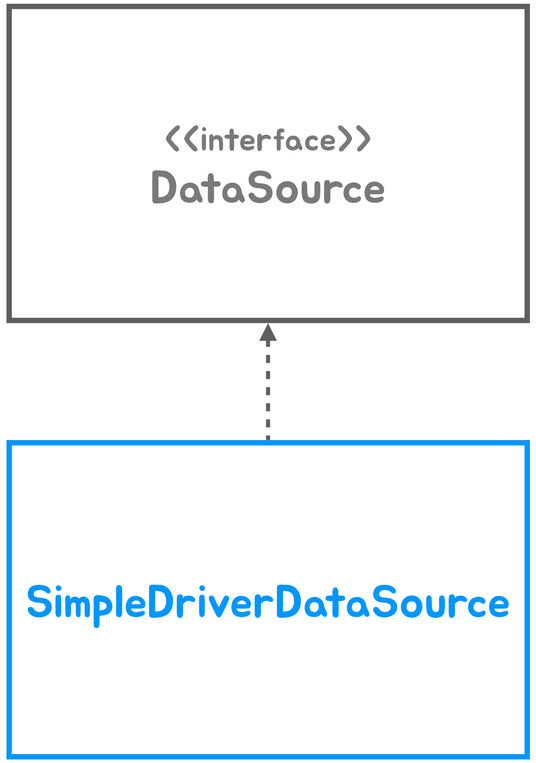
DataSource와 JdbcTemplate

@MyAutoConfiguration DataSourceConfig

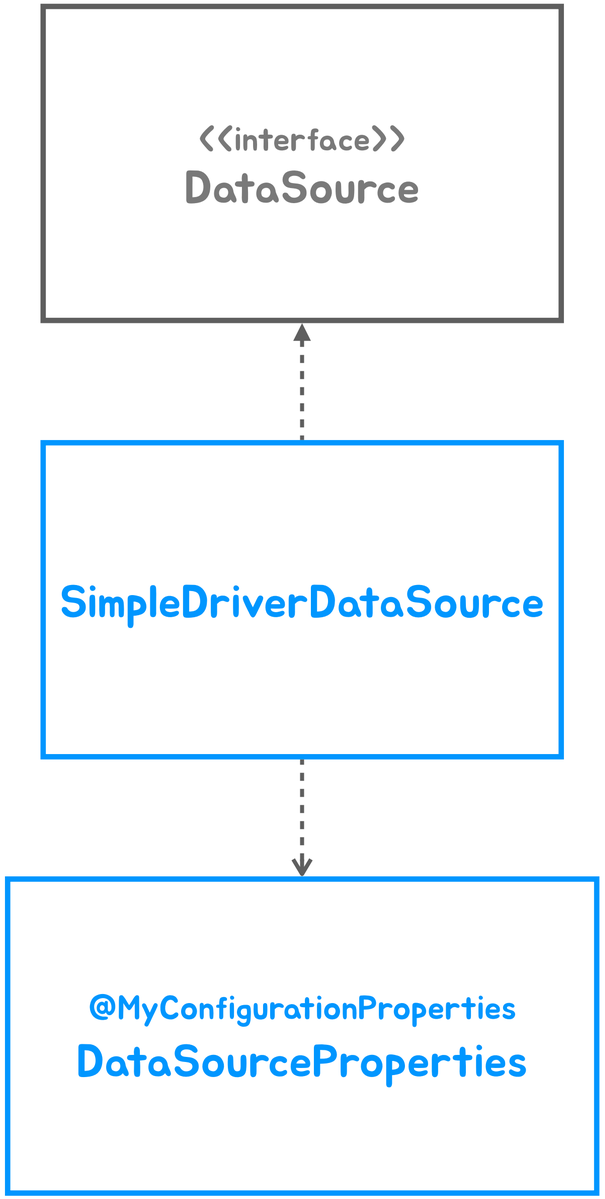
@MyAutoConfiguration DataSourceConfig org.springframework.jdbc.core.JdbcOperations

@MyAutoConfiguration DataSourceConfig org.springframework.jdbc.core.JdbcOperations

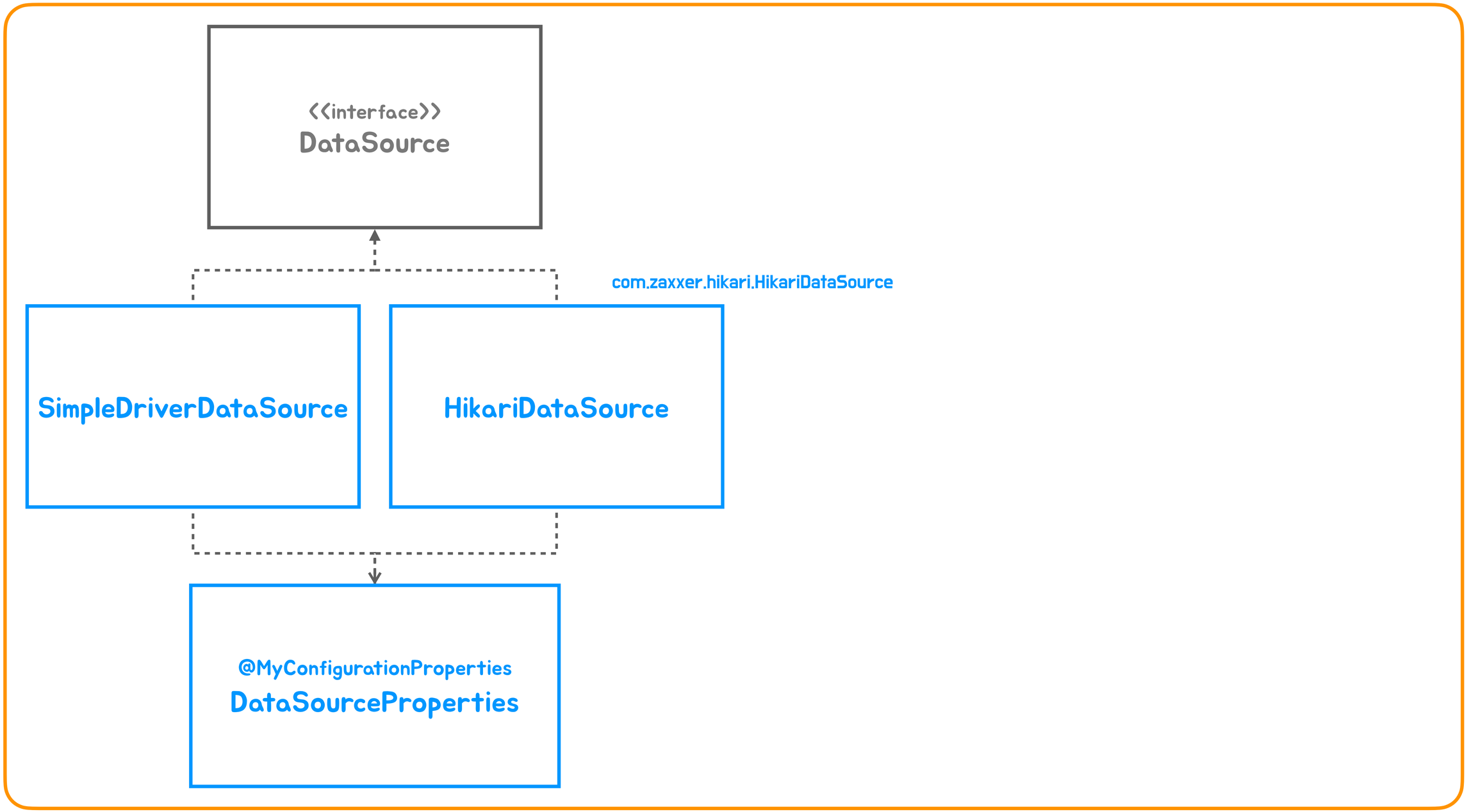
@MyAutoConfiguration DataSourceConfig org.springframework.jdbc.core.JdbcOperations



@MyAutoConfiguration DataSourceConfig org.springframework.jdbc.core.JdbcOperations

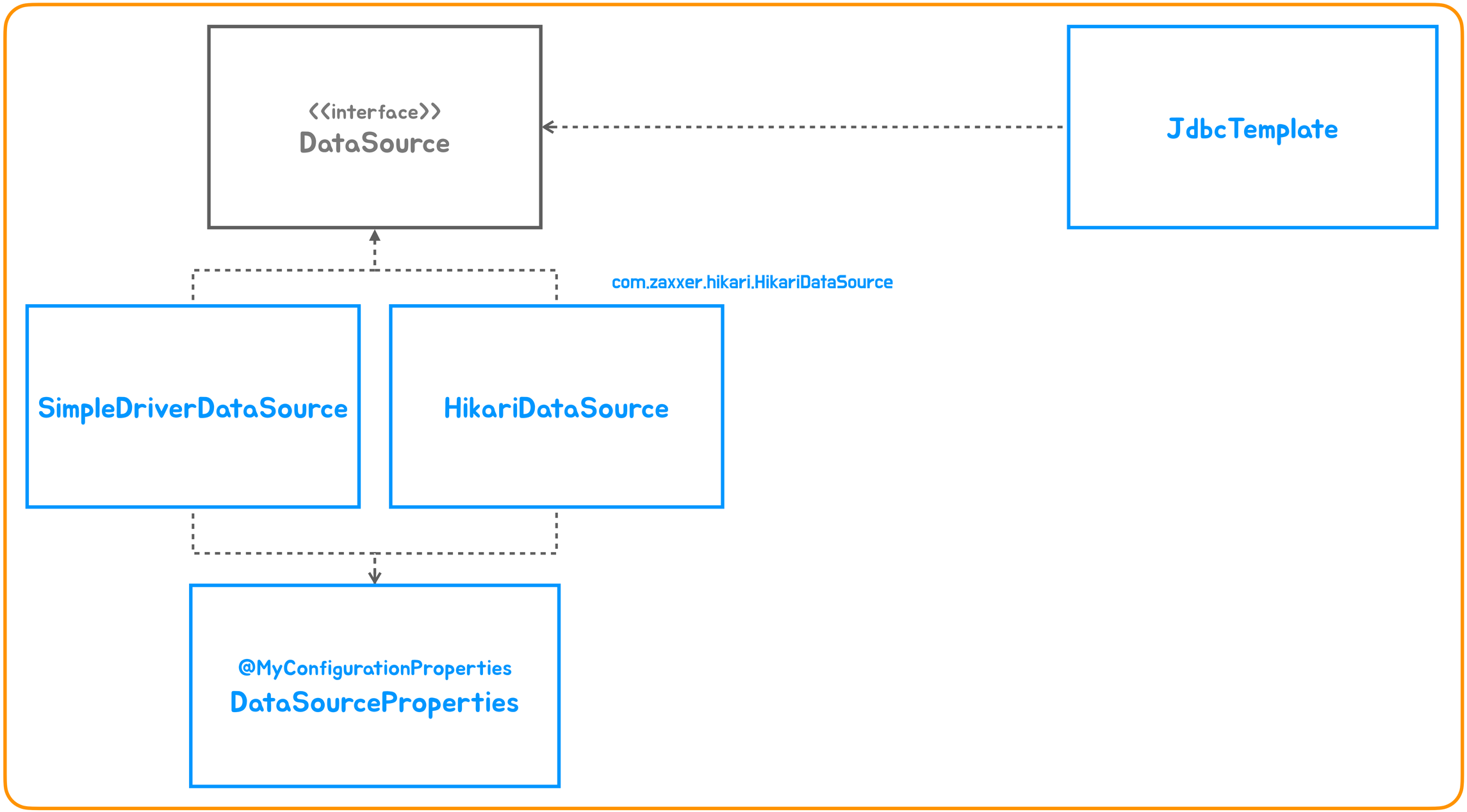


@MyAutoConfiguration DataSourceConfig org.springframework.jdbc.core.JdbcOperations

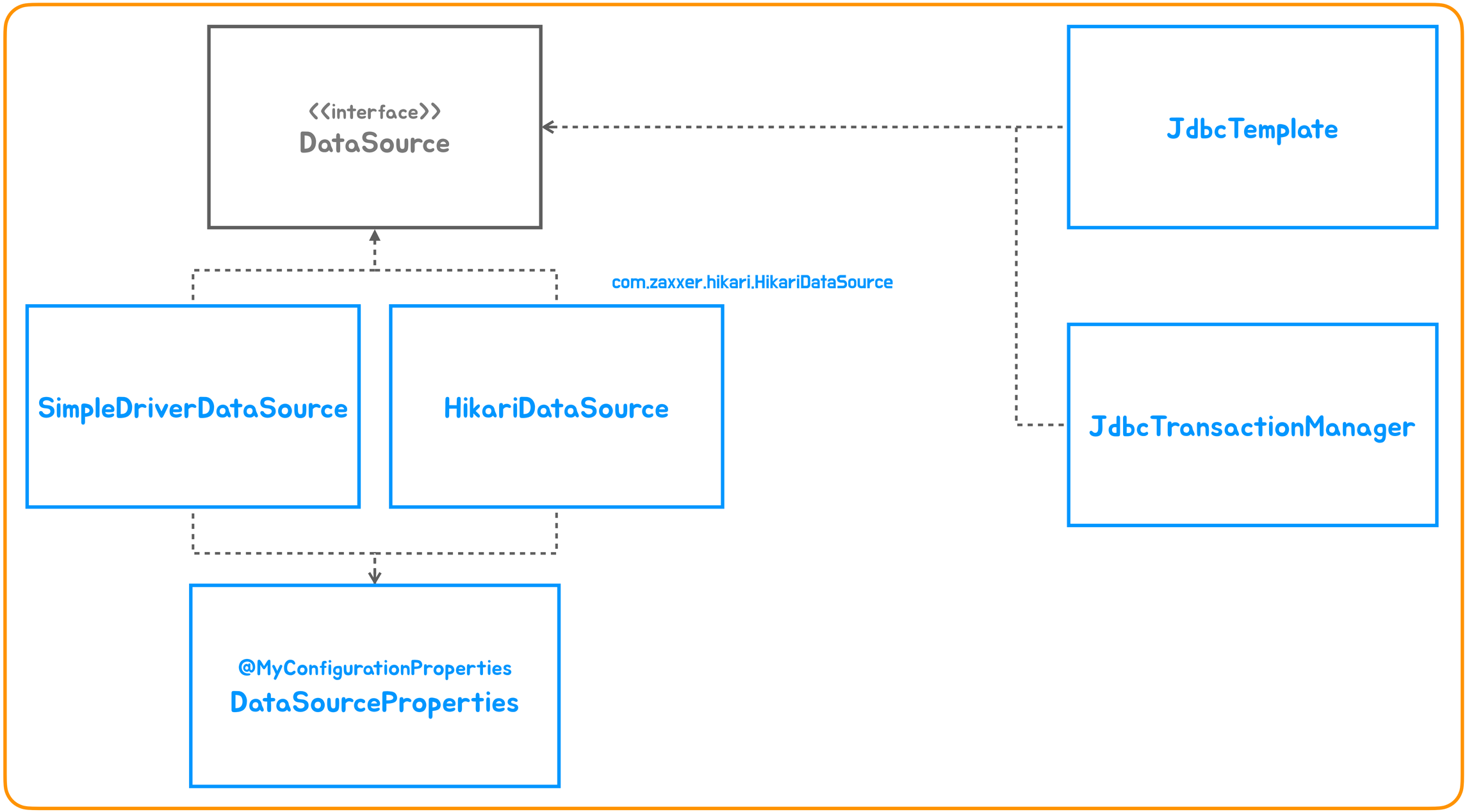


@MyAutoConfiguration DataSourceConfig

org.springframework.jdbc.core.JdbcOperations

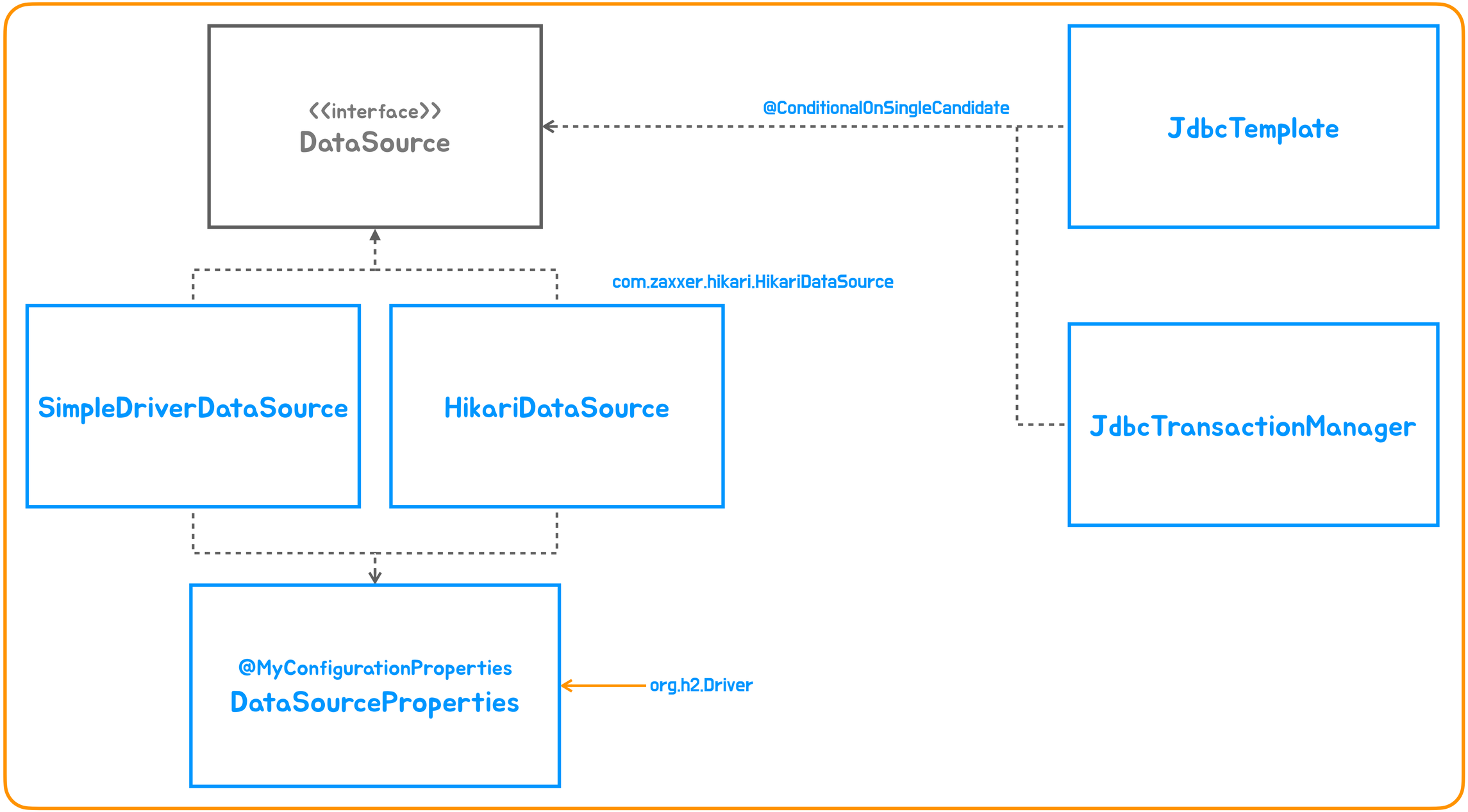


@MyAutoConfiguration DataSourceConfig org.springframework.jdbc.core.JdbcOperations

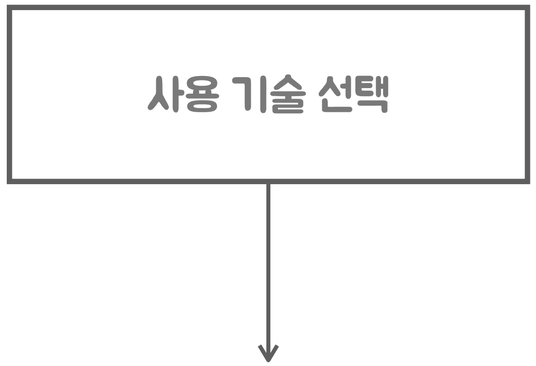


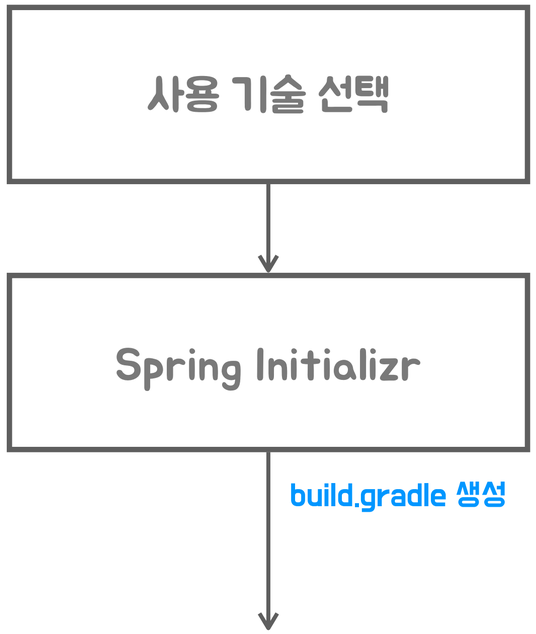
@MyAutoConfiguration DataSourceConfig

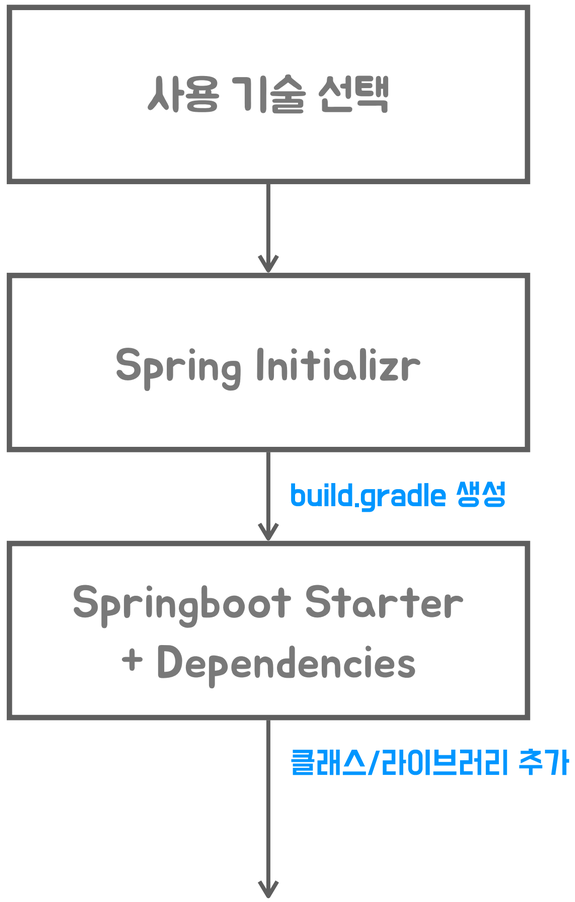
org.springframework.jdbc.core.JdbcOperations

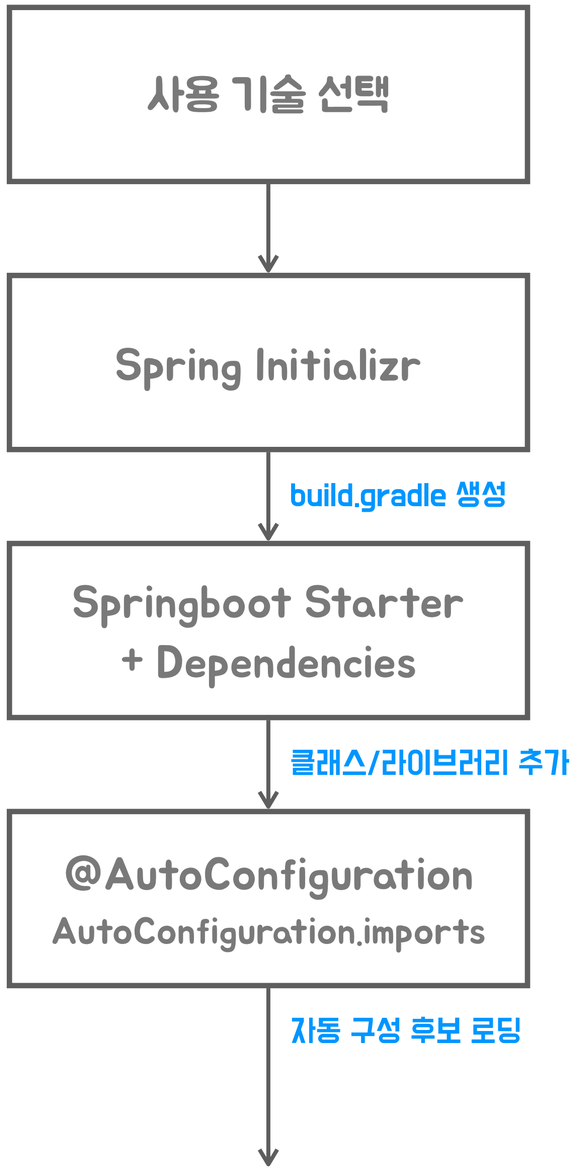


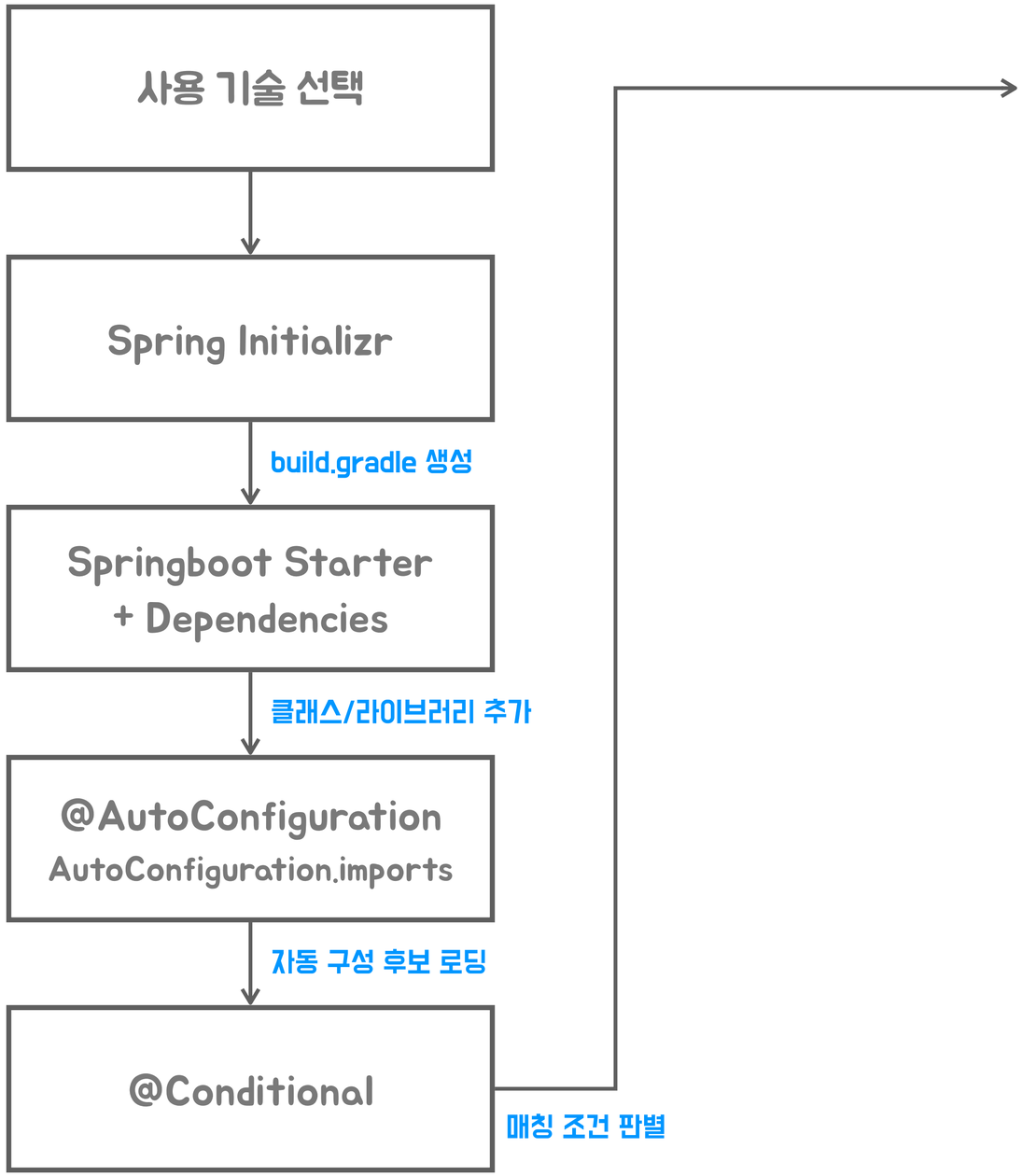
스프링 부트 자세히 살펴보기

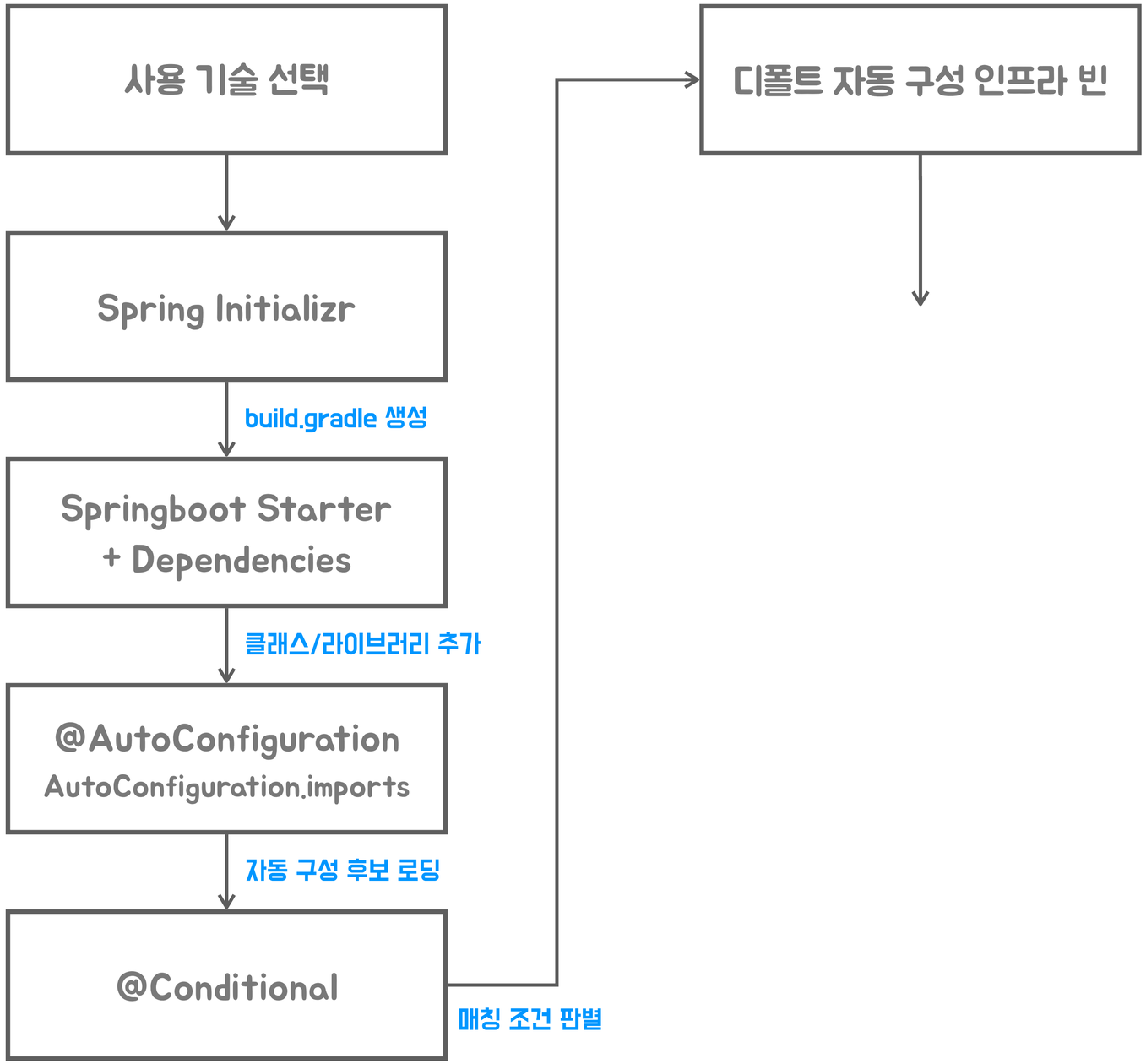

사용 기술 선택

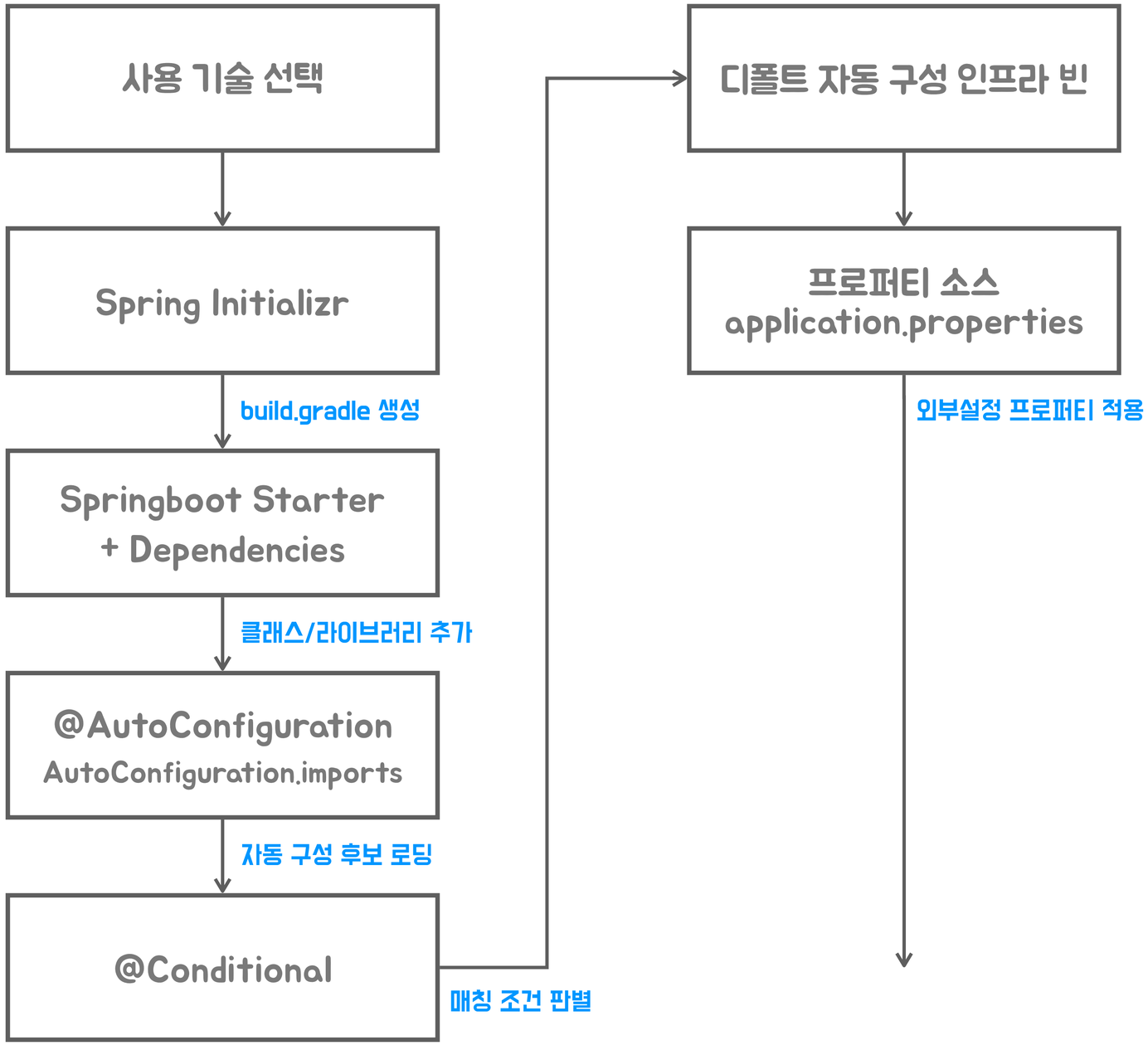

사용 기술 선택
Spring Initializr
build.gradle 생성

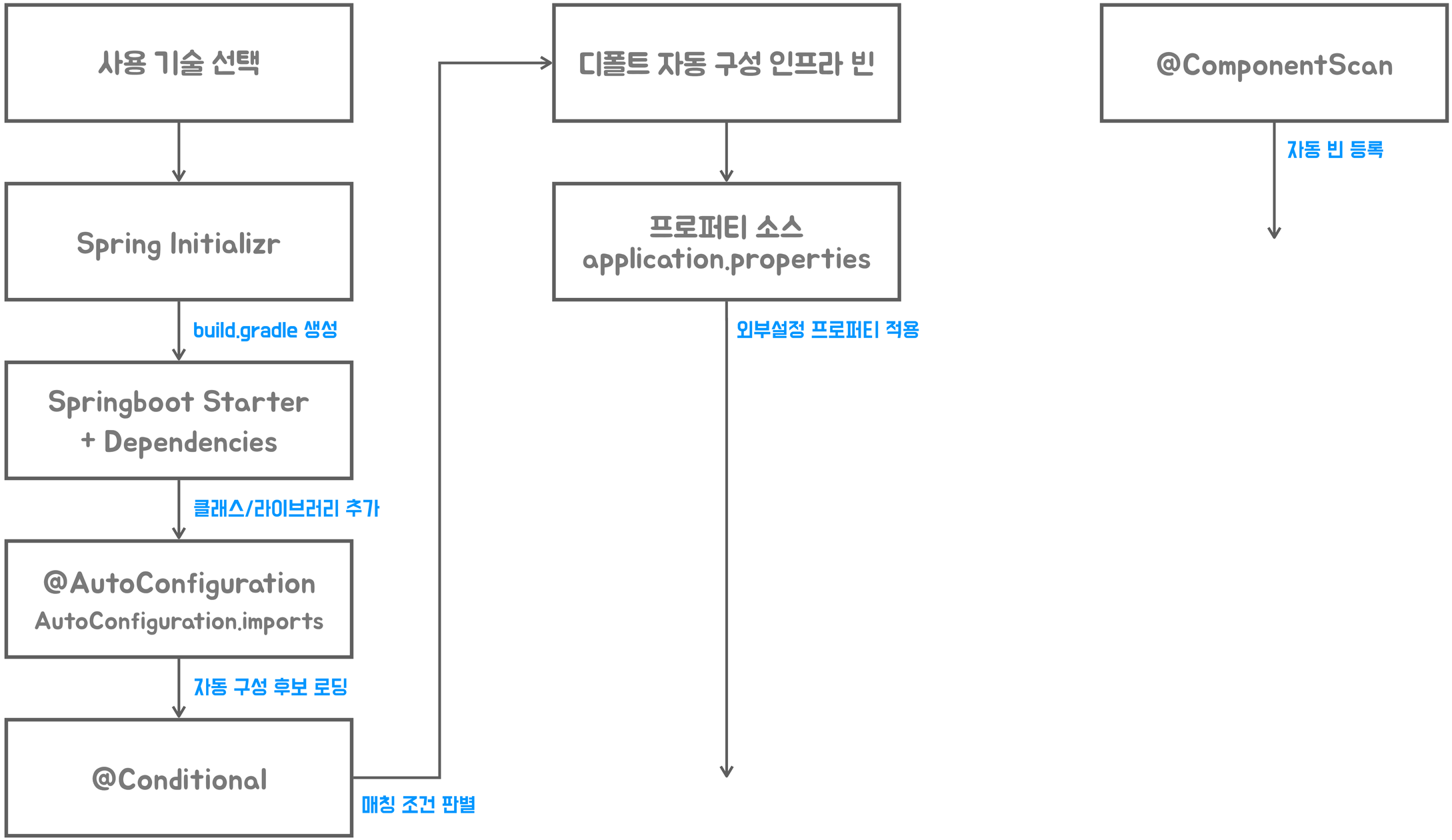

사용 기술 선택
Spring Initializr
build.gradle 생성
Springboot Starter
+ Dependencies
클래스/라이브러리 추가

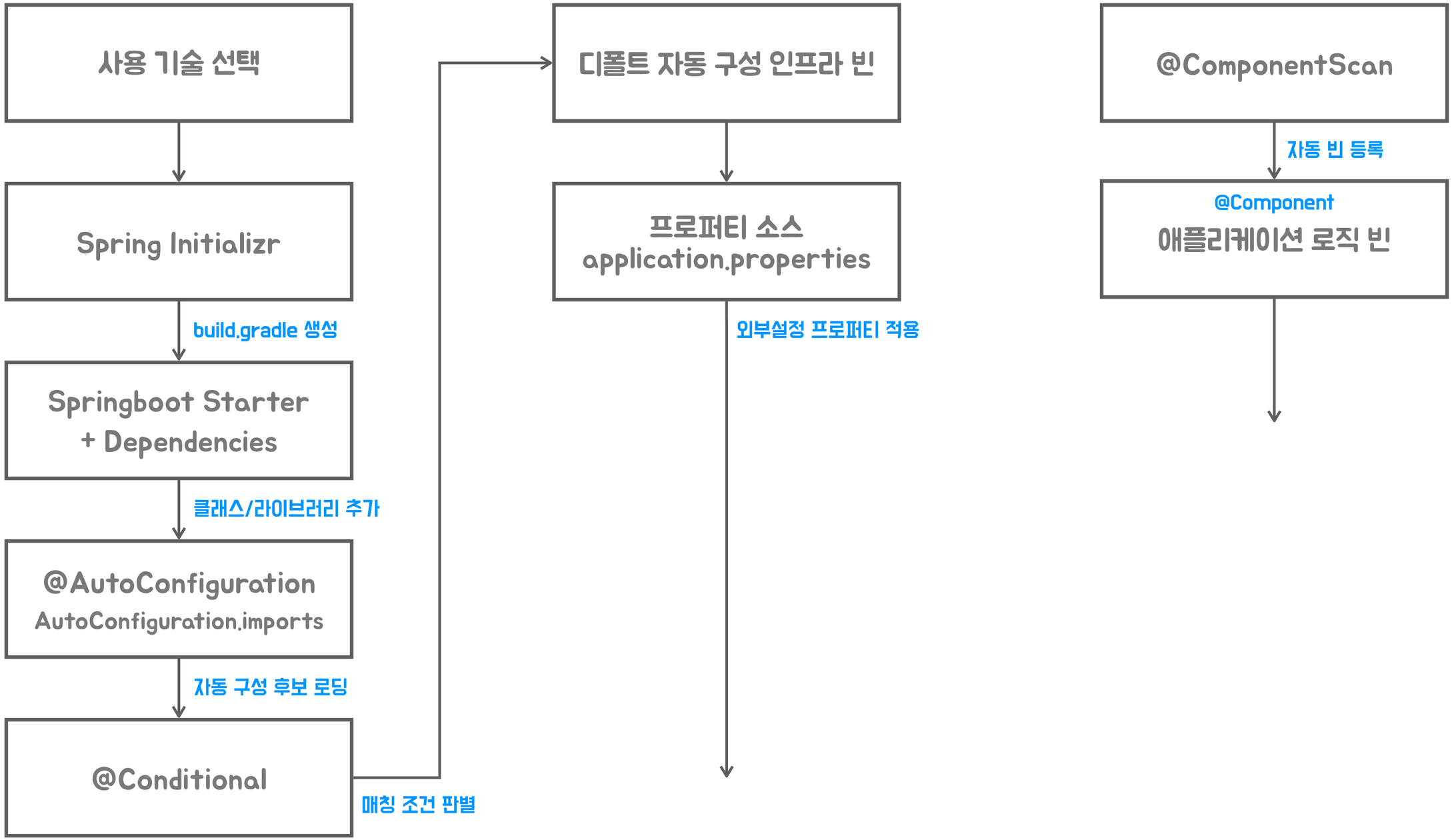

사용 기술 선택
Spring Initializr
build.gradle 생성
Springboot Starter
+ Dependencies
클래스/라이브러리 추가
@AutoConfiguration
AutoConfiguration.imports
자동 구성 후보 로딩

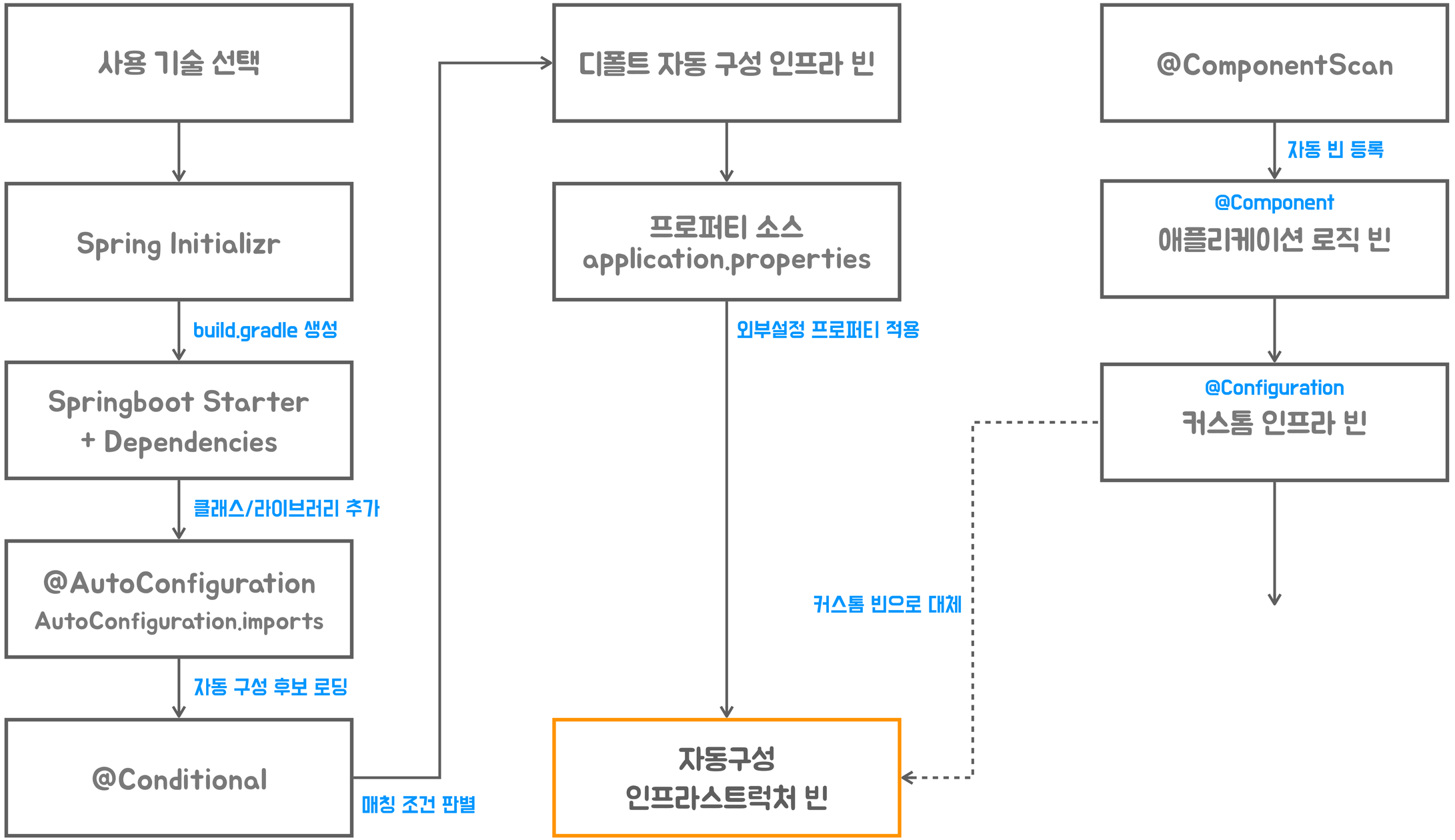

사용 기술 선택
Spring Initializr
build.gradle 생성
Springboot Starter
+ Dependencies
클래스/라이브러리 추가
@AutoConfiguration
AutoConfiguration.imports
자동 구성 후보 로딩
@Conditional
매칭 조건 판별

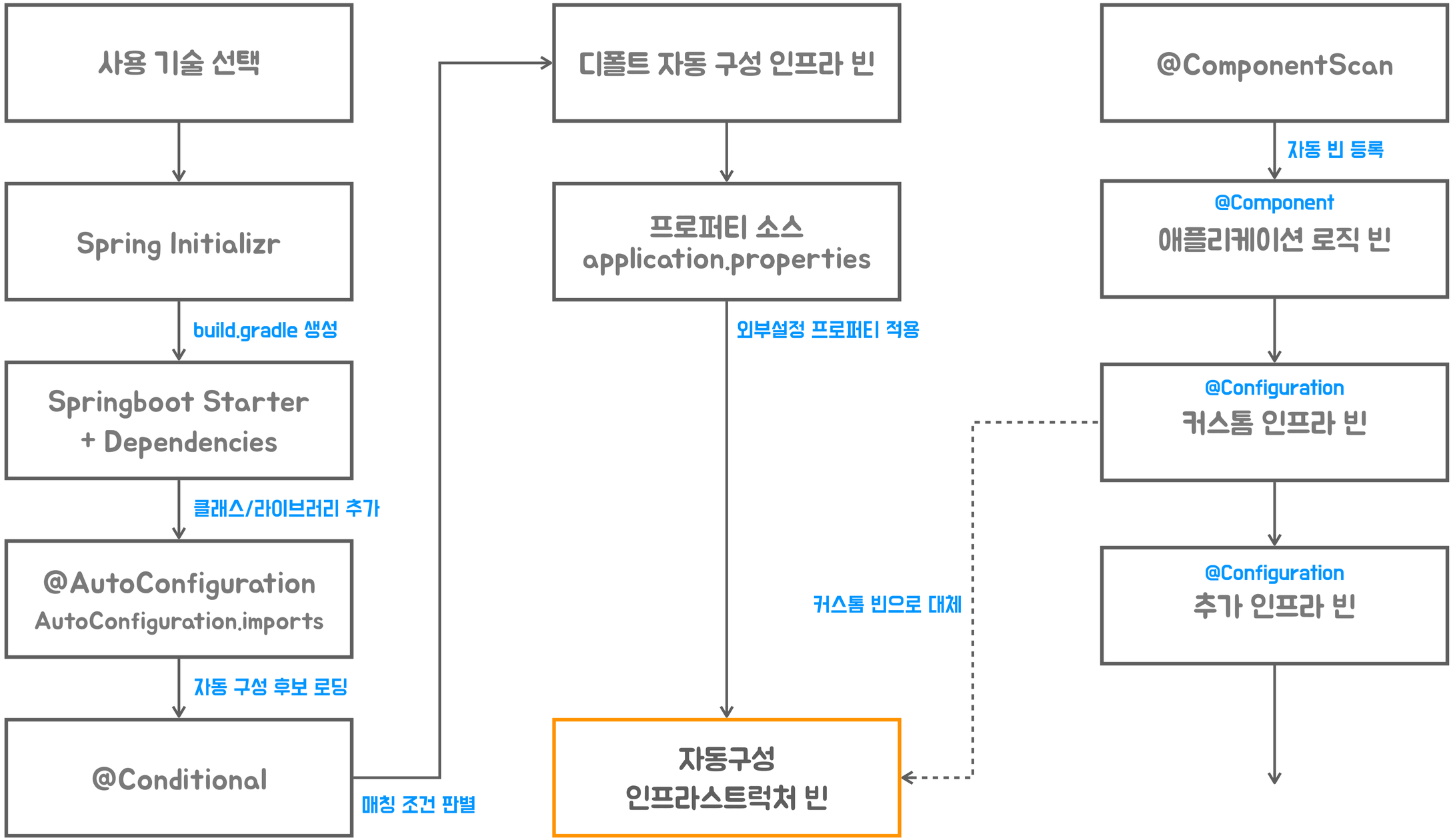

사용 기술 선택
Spring Initializr
build.gradle 생성
Springboot Starter
+ Dependencies
클래스/라이브러리 추가
@AutoConfiguration
AutoConfiguration.imports
자동 구성 후보 로딩
@Conditional
매칭 조건 판별
디폴트 자동 구성 인프라 빈

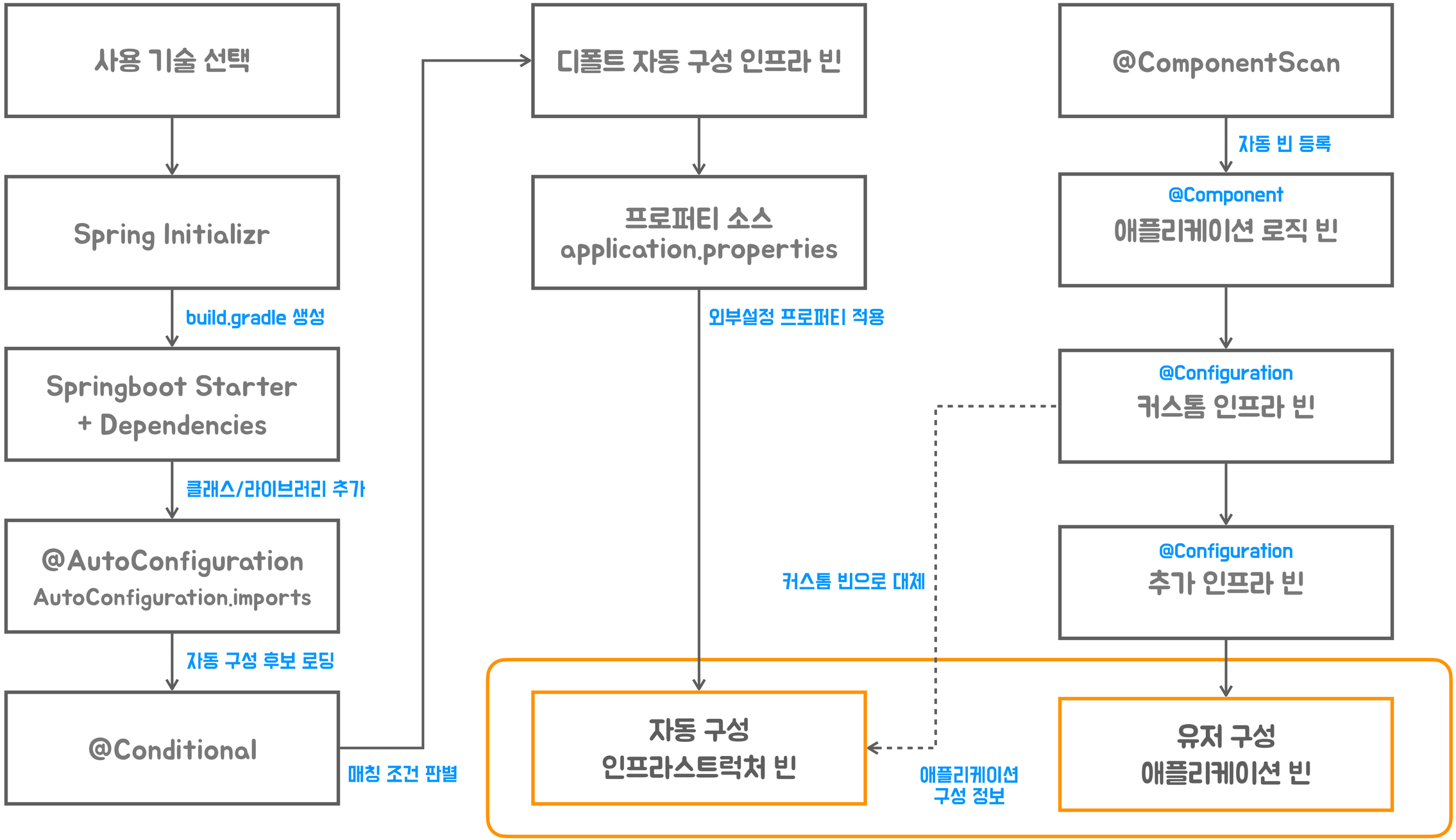

사용 기술 선택
Spring Initializr
build.gradle 생성
Springboot Starter
+ Dependencies
클래스/라이브러리 추가
@AutoConfiguration
AutoConfiguration.imports
자동 구성 후보 로딩
@Conditional
매칭 조건 판별
디폴트 자동 구성 인프라 빈
프로퍼티 소스
application.properties
외부설정 프로퍼티 적용

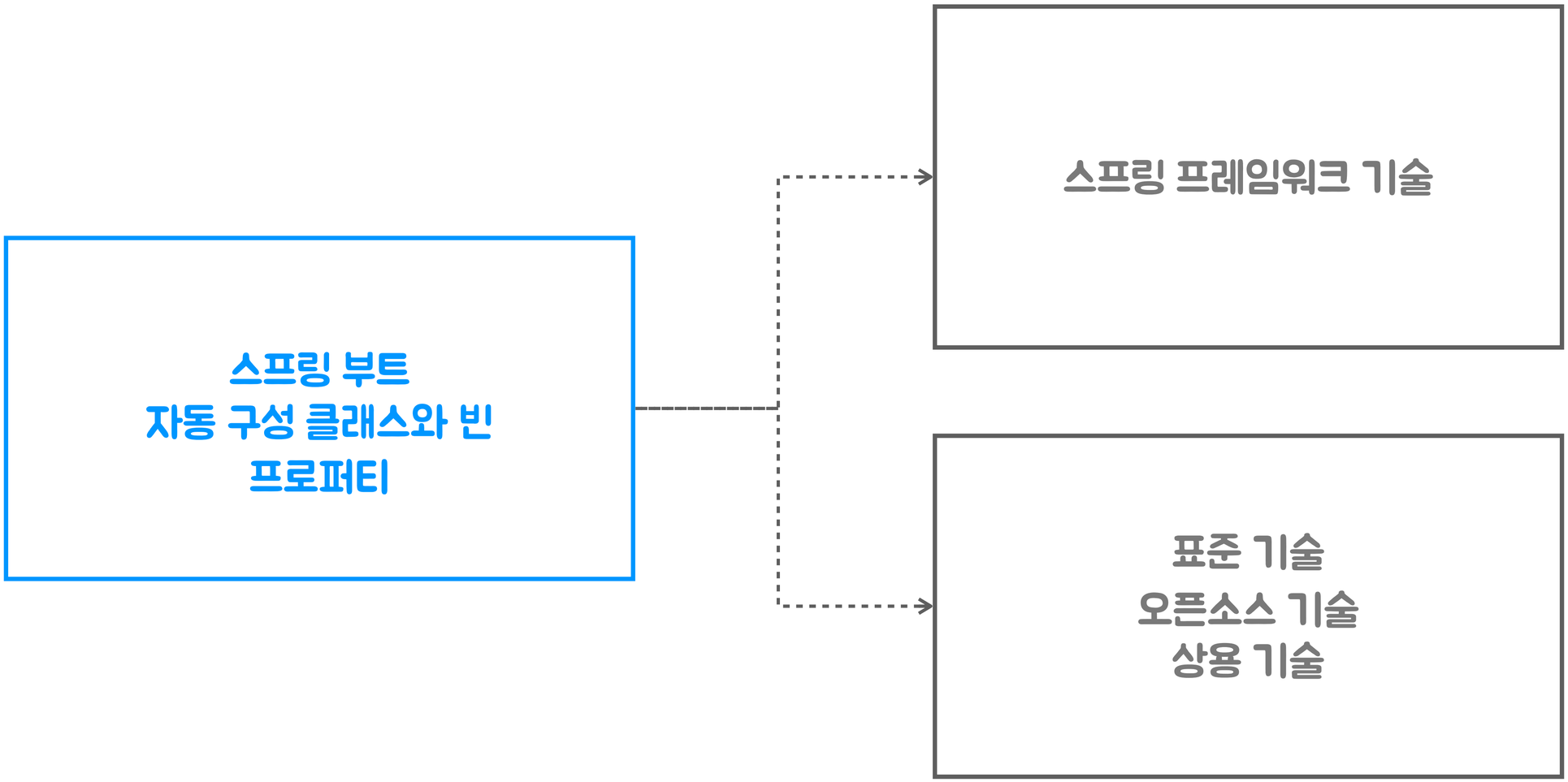

사용 기술 선택
Spring Initializr
build.gradle 생성
Springboot Starter
+ Dependencies
클래스/라이브러리 추가
@AutoConfiguration
AutoConfiguration.imports
자동 구성 후보 로딩
@Conditional
매칭 조건 판별
디폴트 자동 구성 인프라 빈
프로퍼티 소스
application.properties
외부설정 프로퍼티 적용
@ComponentScan
자동 빈 등록


사용 기술 선택
Spring Initializr
build.gradle 생성
Springboot Starter
+ Dependencies
클래스/라이브러리 추가
@AutoConfiguration
AutoConfiguration.imports
자동 구성 후보 로딩
@Conditional
매칭 조건 판별
디폴트 자동 구성 인프라 빈
프로퍼티 소스
application.properties
외부설정 프로퍼티 적용
@ComponentScan
자동 빈 등록
@Component
애플리케이션 로직 빈


사용 기술 선택
Spring Initializr
build.gradle 생성
Springboot Starter
+ Dependencies
클래스/라이브러리 추가
@AutoConfiguration
AutoConfiguration.imports
자동 구성 후보 로딩
@Conditional
매칭 조건 판별
디폴트 자동 구성 인프라 빈
프로퍼티 소스
application.properties
외부설정 프로퍼티 적용
자동구성
인프라스트럭처 빈
커스톰 빈으로 대체
@ComponentScan
자동 빈 등록
@Component
애플리케이션 로직 빈
@Configuration
커스톰 인프라 빈


사용 기술 선택
Spring Initializr
build.gradle 생성
Springboot Starter
+ Dependencies
클래스/라이브러리 추가
@AutoConfiguration
AutoConfiguration.imports
자동 구성 후보 로딩
@Conditional
매칭 조건 판별
디폴트 자동 구성 인프라 빈
프로퍼티 소스
application.properties
외부설정 프로퍼티 적용
자동구성
인프라스트럭처 빈
커스톰 빈으로 대체
@ComponentScan
자동 빈 등록
@Component
애플리케이션 로직 빈
@Configuration
커스톰 인프라 빈
@Configuration
추가 인프라 빈


사용 기술 선택
Spring Initializr
build.gradle 생성
Springboot Starter
+ Dependencies
클래스/라이브러리 추가
@AutoConfiguration
AutoConfiguration.imports
자동 구성 후보 로딩
@Conditional
매칭 조건 판별
디폴트 자동 구성 인프라 빈
프로퍼티 소스
application.properties
외부설정 프로퍼티 적용
@ComponentScan
자동 빈 등록
@Component
애플리케이션 로직 빈
@Configuration
커스톰 인프라 빈
@Configuration
추가 인프라 빈
커스톰 빈으로 대체
자동 구성
인프라스트럭처 빈
애플리케이션
구성 정보
유저 구성
애플리케이션 빈


스프링 부트
자동 구성 클래스와 빈
프로퍼티
스프링 프레임워크 기술
표준 기술
오픈소스 기술
상용 기술

-Ddebug, --debug

자동구성 클래스 Conditon 결과 로그

SpringBoot Reference

문서에서 관련 기술, 자동구성, 프로퍼티 확인

ConditionEvaluationReport

자동구성 클래스 Conditon 결과 빈

@AutoConfiguration
@Conditional
Condition
@Bean

자동 구성 클래스와 조건, 빈 확인

다른 적용 가능한 관련
기술은 뭐가 있을까?

ListableBeanFactory

등록된 빈 확인

Properties
Bind
Customizer
Configurer

프로퍼티 클래스와 바인딩.

정리

스프링 부트는

- 스프링 프레임워크를 잘 쓰게 도와주는 도구의 모음
- 서블릿 컨테이너와 관련된 모든 번거로운 작업을 감춰줌
- 스프링과 각종 기술의 주요 인프라스트럭처 빈을 자동 구성을 이용해서 자동으로 등록해줌
- 외부 설정, 커스톰 빈 등록을 통해서 유연하게 확장 가능

스프링 프레임워크

- 빈 오브젝트의 생명주기를 관리하는 컨테이너
- 빈 오브젝트의 의존 관계를 동적으로 주입해주는 어셈블러
- 구성 정보(configuration metadata)와 애플리케이션 기능을 담은 오브젝트가 결합되어 동작하는 애플리케이션이 된다
- @Configuration, @Bean, @Import를 이용한 구성 정보
- 메타 애노테이션, 합성 애노테이션 활용

스프링 부트의 이해

- 스프링 부트가 스프링의 기술을 어떻게 활용하는지 배우고 응용할 수 있다
- 스프링 부트가 선택한 기술, 자동으로 만들어주는 구성, 디폴트 설정이 어떤 것인지 확인할 수 있다
- 필요할 때 부트의 기본 구성을 수정하거나, 확장할 수 있다
- 나만의 스프링 부트 모듈을 만들어 활용할 수 있다

강의의 목표

- 스프링 부트로 만든 스프링 애플리케이션의 기술과 구성 정보를 직접 확인할 수 있다
- 적용 가능한 설정 항목을 파악할 수 있다
- 직접 만든 빈 구성 정보를 적용하고, 그에 따른 변화를 분석할 수 있다
- 스프링 부트의 기술을 꼼꼼히 살펴볼 수 있다

스프링 부트 더 알아가기

- 스프링 부트의 코어 (Profile, Logging, Testing...)
- 핵심 기술 영역 (Web, Data, Messaging, IO...)
- 운영환경의 모니터링, 관리 방법
- 컨테이너, 배포, 빌드 툴
- 스프링 부트 3.x
- 스프링 프레임워크와 자바 표준, 오픈소스 기술

SpringBoot 3.0 업그레이드

스프링 부트 3.0

- Spring 6
- JDK 17 또는 그 이상
- Jakarta EE 9, 10

2.7 예제 3.0으로 업그레이드

- build.gradle의 스프링 부트 버전 수정
- Jakarta EE의 패키지명으로 변경
- SpringMVC의 바뀐 동작 방식 적용 - 타입 레벨 단독 @RequestMapping 문제

build.gradle

```
plugins {
    id 'java'
    id 'org.springframework.boot' version '3.0.1'
    id 'io.spring.dependency-management' version '1.1.0'
}

group = 'tobyspring'
version = '0.0.1-SNAPSHOT'
sourceCompatibility = '17'
```

Jakarta EE 패키지명으로 변경

```
import jakarta.annotation.PostConstruct;


import jakarta.servlet.ServletException;

import jakarta.servlet.http.HttpServlet;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;
```


JAKARTA EE 9 PLATFORM
JAKARTA EE 9 WEB PROFILE
Concurrency 2.0
Authorization 2.0
Authentication 2.0
Server Pages 3.0
Persistence 3.0
Activation 2.0
CDI 3.0
WebSocket 2.0
Restful Web Services 3.0
XML Binding 3.0 *
Batch 2.0
Expression Language 4.0
Bean Validation 3.0
JSON Processing 2.0
Enterprise Web Services 2.0 *
Connectors 2.0
Faces 3.0
Debugging Support 2.0
JSON Binding 2.0
XML Web Services 3.0 *
Mail 2.0
Security 2.0
Enterprise Beans Lite 4.0
Annotations 2.0
Web Services Metadata 3.0 *
Messaging 3.0
Servlet 5.0
Managed Beans 2.0
Interceptors 2.0
SOAP with Attachments 2.0 *
Enterprise Beans 4.0
Standard Tag Libraries 2.0
Transactions 2.0
Dependency Injections 2.0
* Optional

타입 레벨 @RequestMapping 문제

- Spring 6는 타입 레벨에 단독으로 존재하는 @RequestMapping을 DispatcherServlet이 인식하지 못함. @Controller까지 지정해줘야 한다.
- 섹션 4의 [애노테이션 매핑 정보 사용]의 HelloController에 @Controller까지 추가

감사합니다