코드로 배우는 스프링 웹 프로젝트

PART 1

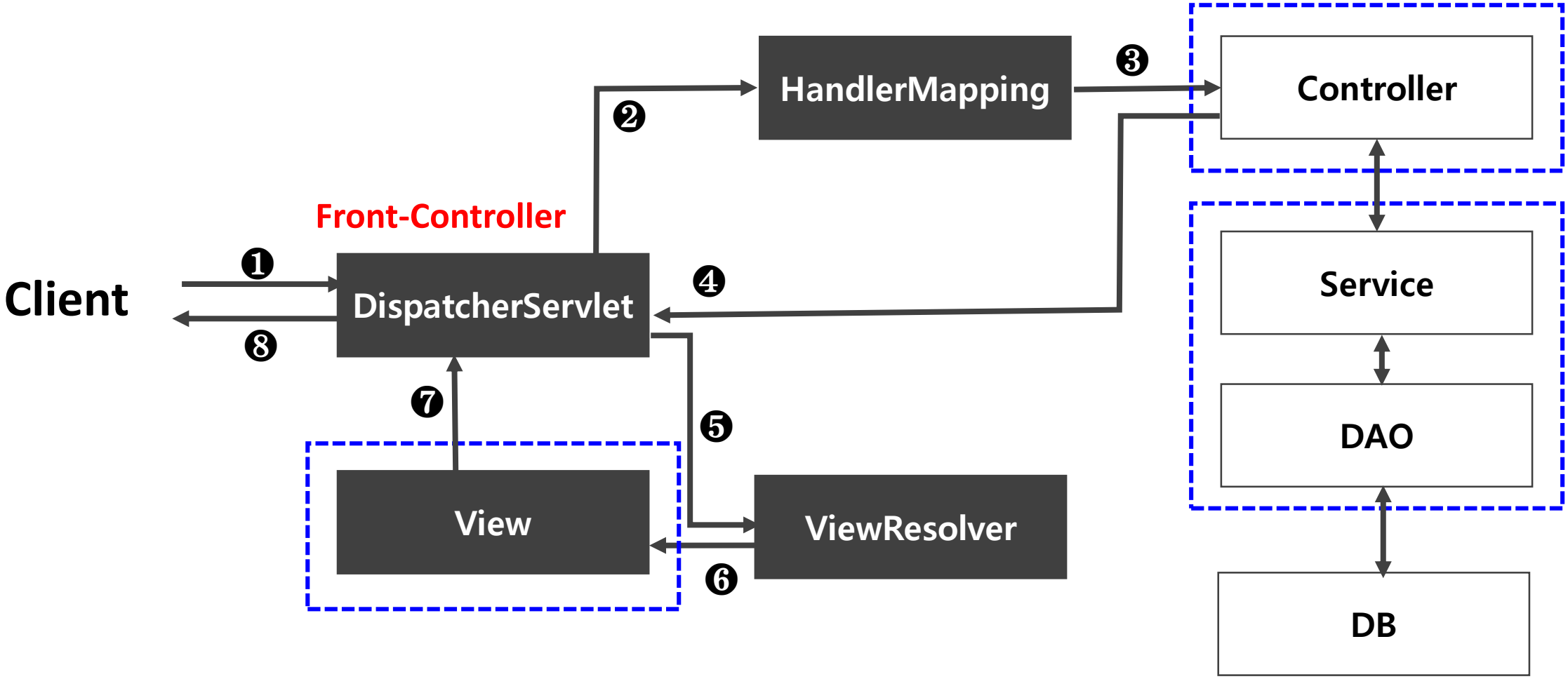
스프링 개발 환경 구축

동의과학대학교
김진숙

2장. 스프링 특징과 의존성 주입

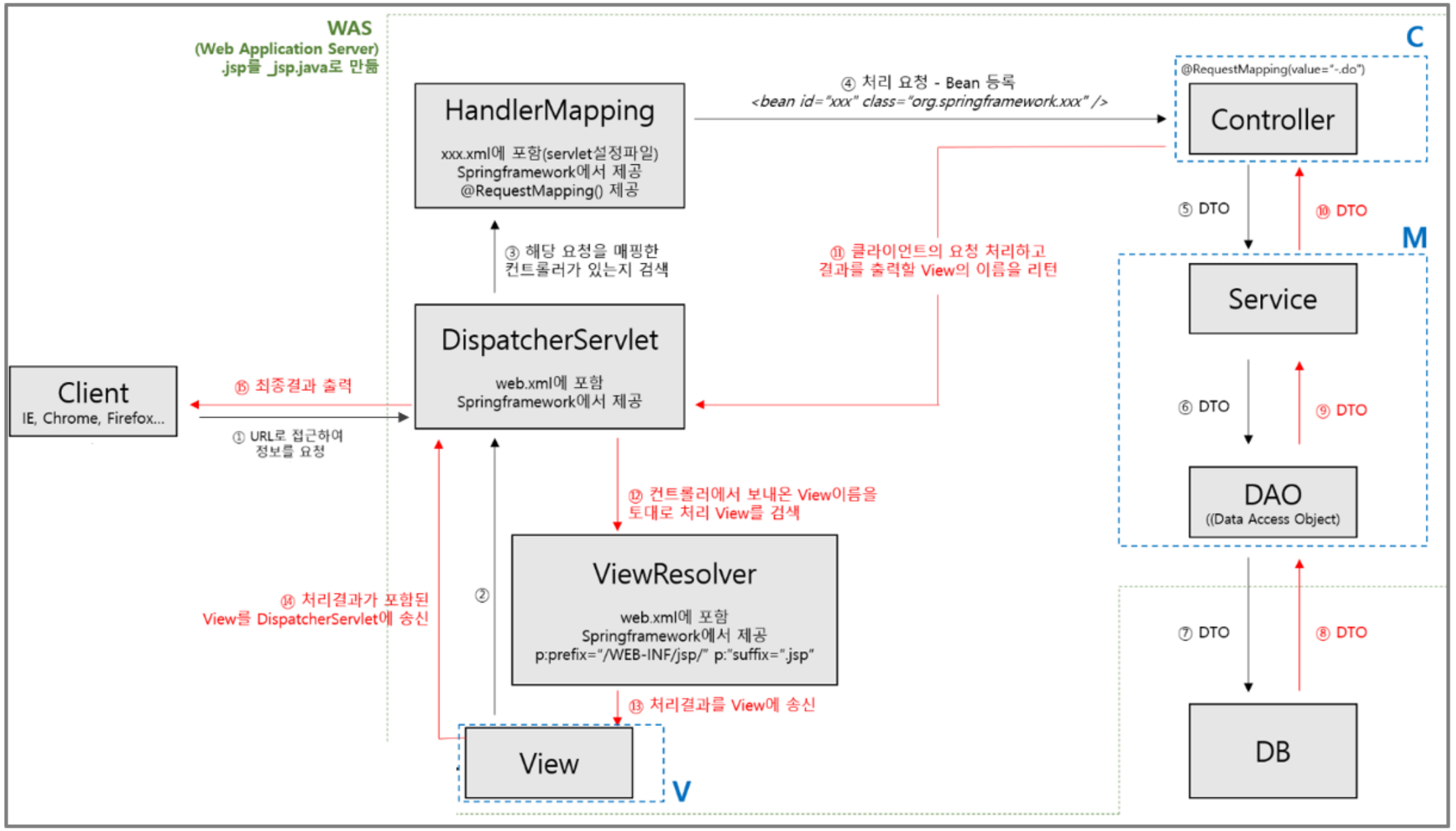
PART 1

스프링 MVC의 기본 흐름(중요)



DispatcherServlet
내부적으로 스프링 컨테이너를 생성

Spring Web MVC 처리 과정



web.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee https://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">

    <!-- 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>

    <!-- Processes application requests -->
    <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>/</url-pattern>
    </servlet-mapping>
</web-app>
```

클라이언트의 모든 요청을 받는 서블릿

스프링 MVC의 기본 흐름(중요)

❶ 사용자 요청(request)은 Front-Controller 인 **DispatcherServlet** 을 통해 처리됨
- 모든 request의 URL을 DispatcherServlet이 받도록 설정되어 있음(web.xml)

❷ **HandlerMapping**은 요청(request) 처리를 담당하는 컨트롤러를 찾기 위해 존재
- @RequestMapping 어노테이션이 적용된 것을 기준으로 판단

❸ 적절한 컨트롤러를 찾으면 **HandlerAdaptor**를 이용하여 해당 컨트롤러를 동작 시킴

❹ Controller는 개발자가 작성하는 클래스로 실제 요청(request)을 처리하는 로직 작성
- View에 전달해야 하는 데이터는 주로 **Model** 객체에 담아서 전달
- Controller는 다양한 타입의 결과를 반환하는데 이는 **ViewResolver**가 담당함

❺ **ViewResolver**는 Controller가 반환한 결과를 어떤 View를 통해서 처리하는 것이 좋을지 해석하는 역할
- Servlet-context.xml에 정의된 InternalResourceViewResolver 사용

❻ View는 실제로 응답보내야 하는 데이터를 jsp 등을 이용해 생성하는 역할

❼ 처리 결과가 포함된 View를 DispatcherServlet로 송신

❽ 최종결과가 클라이언트로 전달됨

스프링 프레임워크의 간략한 역사

- 프레임워크란?
  - 뼈대나 근간을 이루는 코드들의 묶음
- 프레임워크를 사용하는 이유
  - 프로그램의 기본 흐름이나 구조를 정하고, 모든 팀원이 이 구조에 자신의 코드를 추가하는 방식으로 개발하기 위한 도구
  - 개발에 필요한 구조를 이미 만들어 놓았기 때문에 반쯤 완성한 상태에서 필요한 부분을 조립하는 형태의 개발 가능
  - 일정한 품질이 보장되는 결과물을 얻을 수 있음
  - 개발 시간 단축도 가능

Heavy / LightWeight Framework

- 2000년대 초반의 분위기
  - 안정된 품질의 개발이 절실
  - EJB(복잡한 엔터프라이즈 프레임워크)
  - 비싼 WAS(web application server)
  - 많은 것의 통합
- 2000년대 중반 이후
  - 빠르고 가벼운 개발 방식
  - 작은 서비스의 군집화

스프링 프레임워크의 차별적 특성

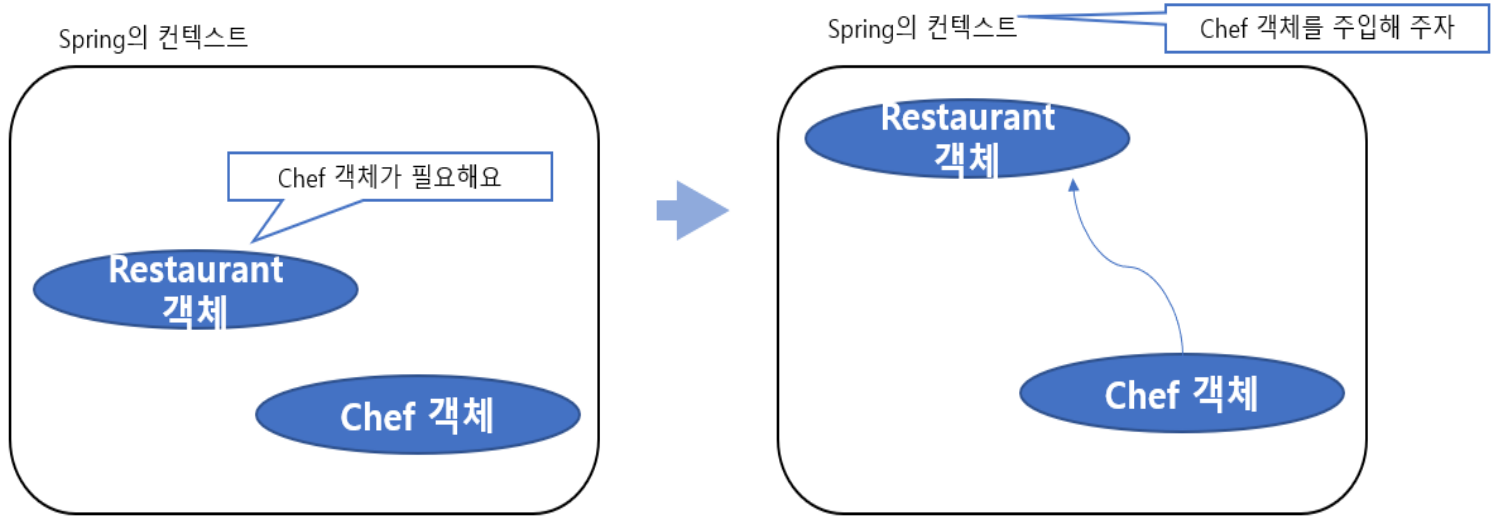
- 복잡함에 반기를 들어 만들어진 객체지향 프레임워크
- 프로젝트 전체 구조를 설계할 때 유용한 프레임워크
- 다른 프레임워크 포용
- 개발 생산성과 개발 도구 지원

스프링의 주요 특징

- **POJO**(Plain Old Java Object) 기반의 구성
  - 일반 자바 객체로 특정 라이브러리나 컨테이너 기술에 종속적이지 않음
- 의존성 주입(**DI**)을 통한 객체 간의 관계 구성
  - 코드에 필요한 객체를 외부에서 제공
- **AOP**(Aspect oriented Programming) 지원
  - 보안, 로그, 트랜잭션과 같이 비즈니스 로직은 아니나 반드시 처리가 필요한 횡단 관심사(cross concern)을 분리하여 제작 지원
- 편리한 **MVC** 구조
- **WAS** 에 종속적이지 않은 개발 환경

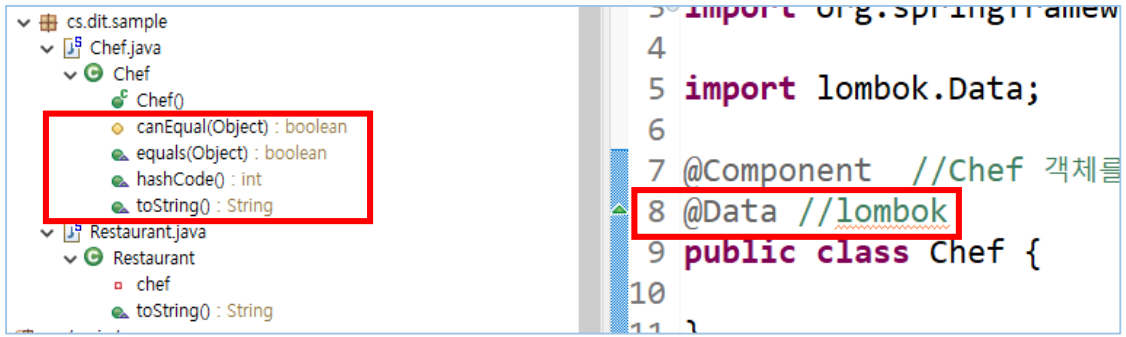
의존성 주입 예제

- 스프링을 이용하는 환경에서 각 각의 객체를 생성하고, 이를 스프링의 설정을 통해서 연결해 보도록 한다.
- 의존성 주입 방법
  - **멤버 변수**를 이용한 DI
  - **Setter 메소드**를 이용한 DI
  - **생성자**를 이용한 DI
  - **Final 필드 자동** 주입



cs.dit.sample.Chef 클래스

```
package cs.dit.sample;

import org.springframework.stereotype.Component;
import lombok.Data;

@Component
@Data
public class Chef {

}
```

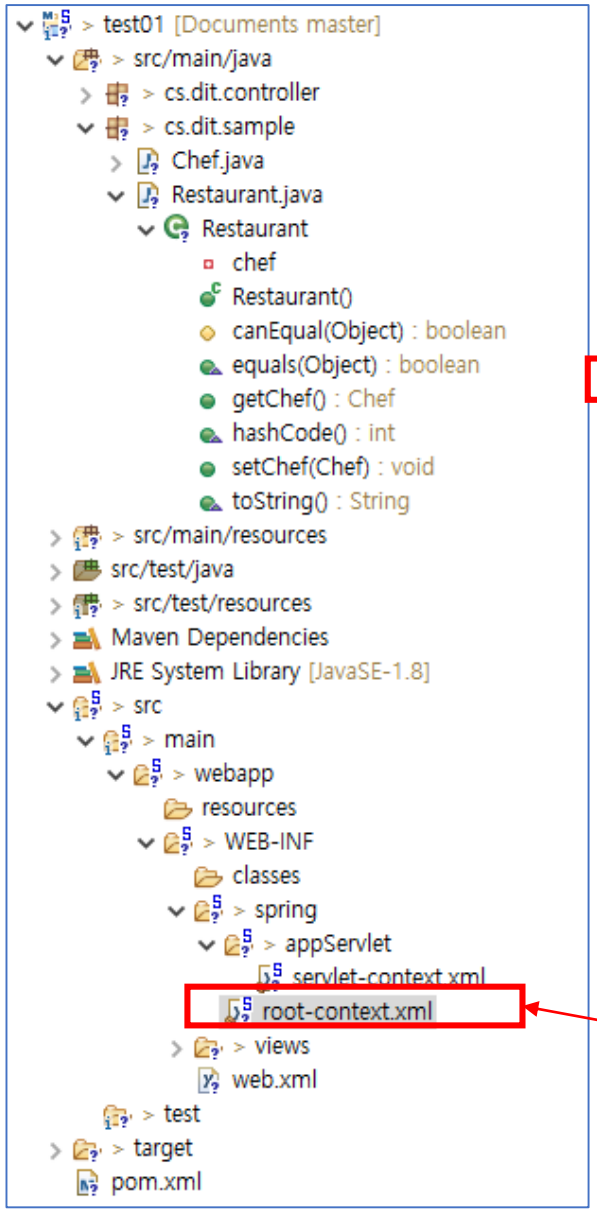


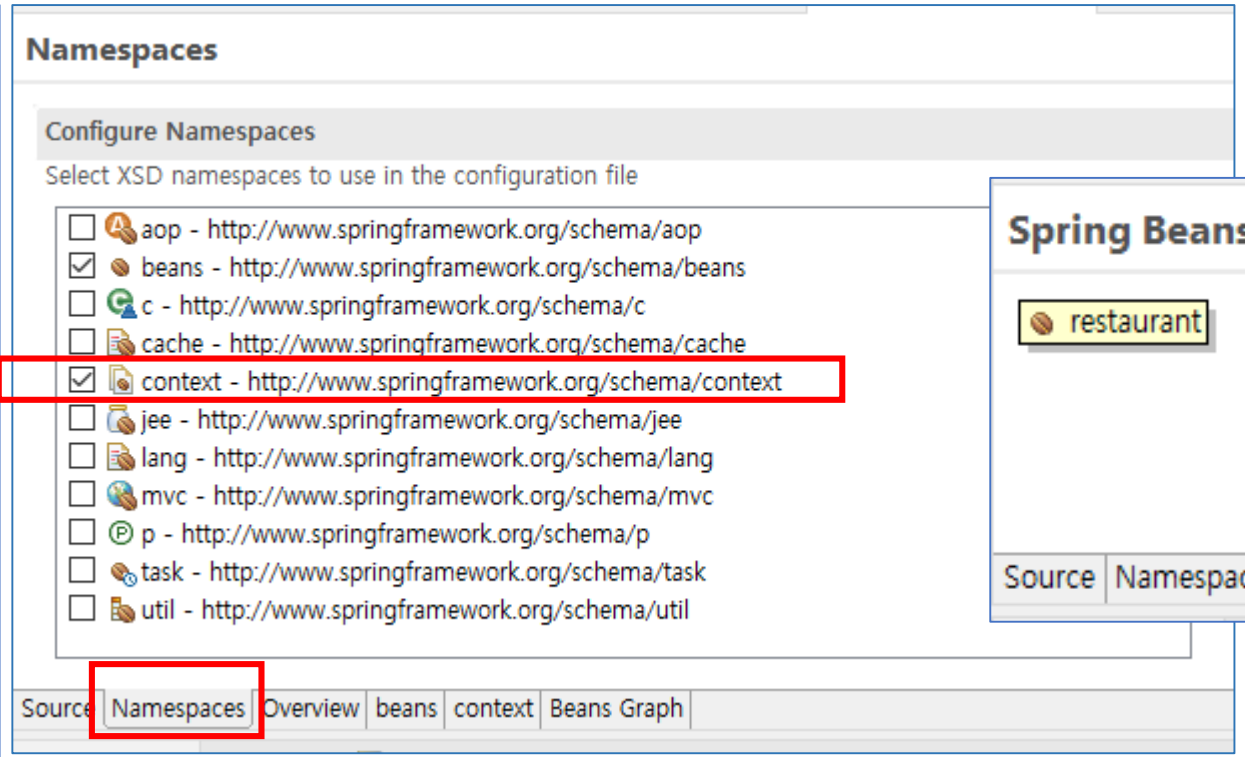
계속사용해야하는 객체는 Bean으로 등록해야 함

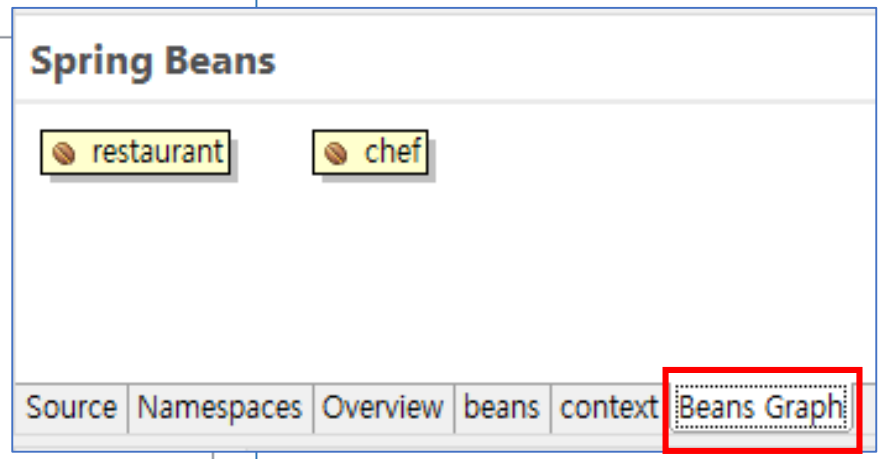
cs.dit.sample.Restaurant 클래스

```
package cs.dit.sample;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;
import lombok.Data;
import lombok.Setter;


@Component
@Data
public class Restaurant {


  @Setter(onMethod_ = {@Autowired})
  private Chef chef;
}
```

Spring Bean 스캔



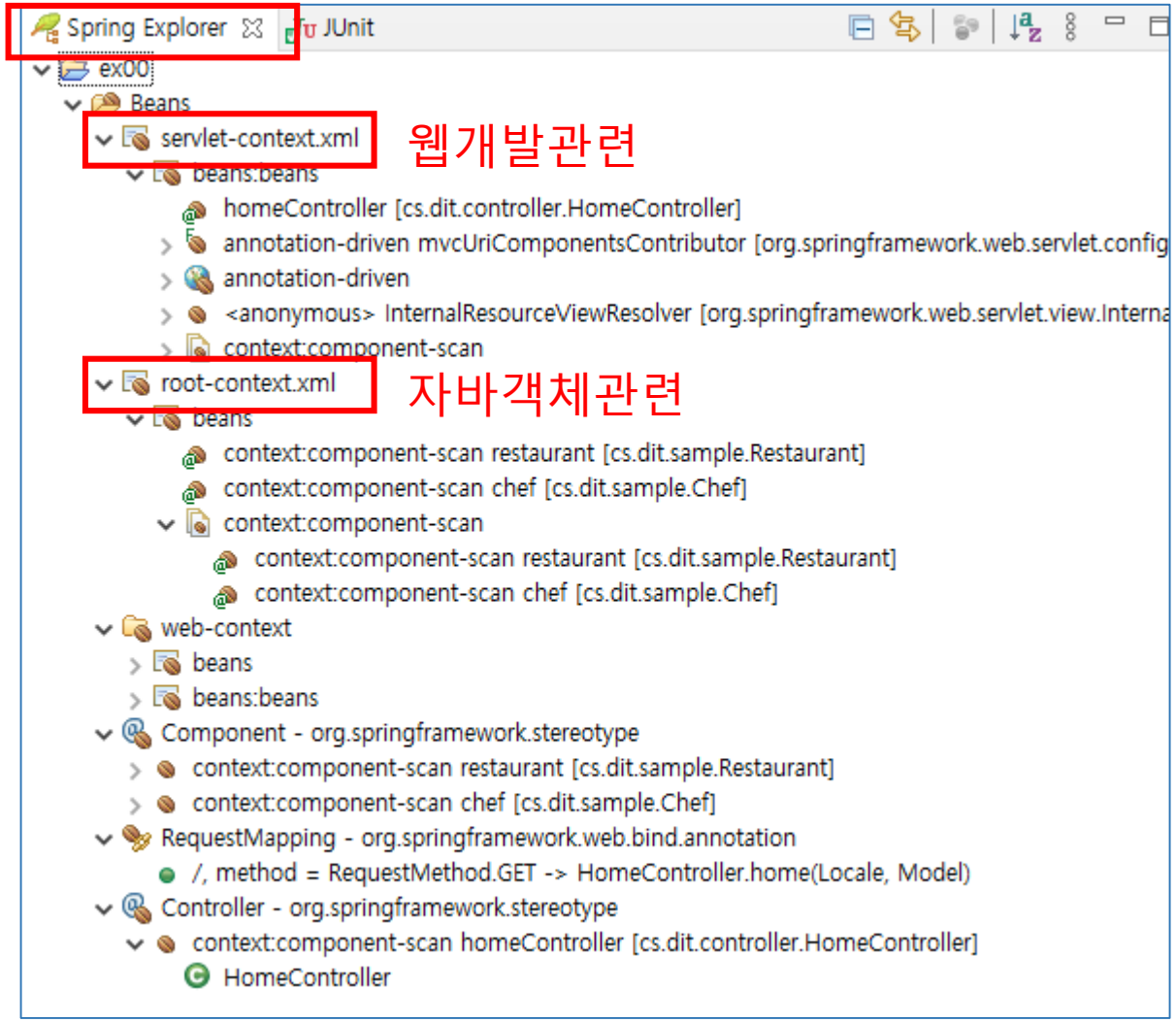




root-context.xml 파일에 추가

```
<context:component-scan base-package="cs.dit.sample"></context:component-scan>
```

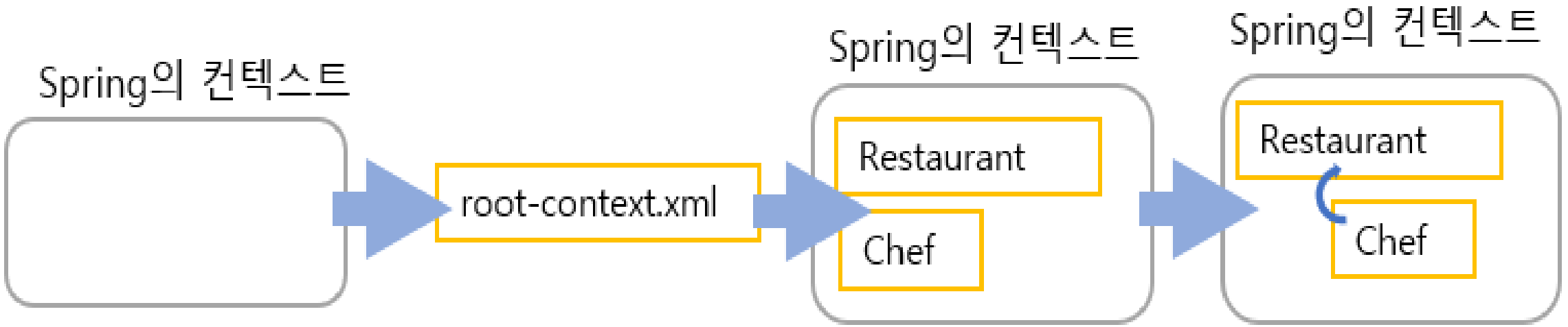
컴포넌트(빈)를 스캔



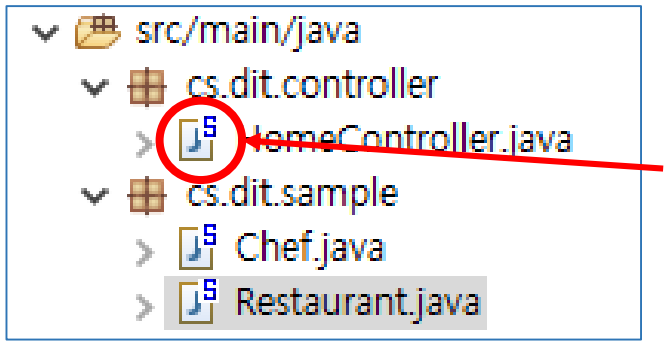
설정파일이 둘로 나누어 정의되는 이유

- Spring은 객체지향 프레임워크
- 웹에 대한 설정을 따로 정의

스프링이 동작하면서 생기는 일



root-context.xml의 설정 내용이 동작하면서 필요한 인스턴스들(beans)을 생성하고, 의존 관계를 파악해서 주입시켜 주는 방식



파란색 s 표시는 스프링이 관리하는 객체라는 의미

Pom.xml 설정

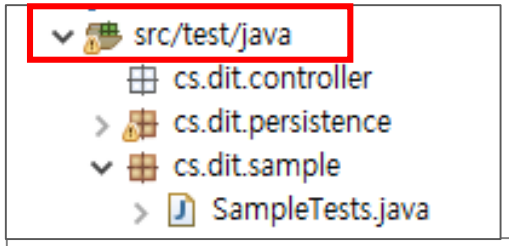
```
<dependency>
    <groupId>log4j</groupId>
    <artifactId>log4j</artifactId>
    <version>1.2.17</version>
    <exclusions>
        <exclusion>
            <groupId>javax.mail</groupId>
            <artifactId>mail</artifactId>
        </exclusion>
        <exclusion>
            <groupId>javax.jms</groupId>
            <artifactId>jms</artifactId>
        </exclusion>
        <exclusion>
            <groupId>com.sun.jdmk</groupId>
            <artifactId>jmxtools</artifactId>
        </exclusion>
        <exclusion>
            <groupId>com.sun.jmx</groupId>
            <artifactId>jmxri</artifactId>
        </exclusion>
    </exclusions>
    <scope>runtime</scope>
</dependency>
```

삭제할 것

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

테스트를 위해 추가할 것

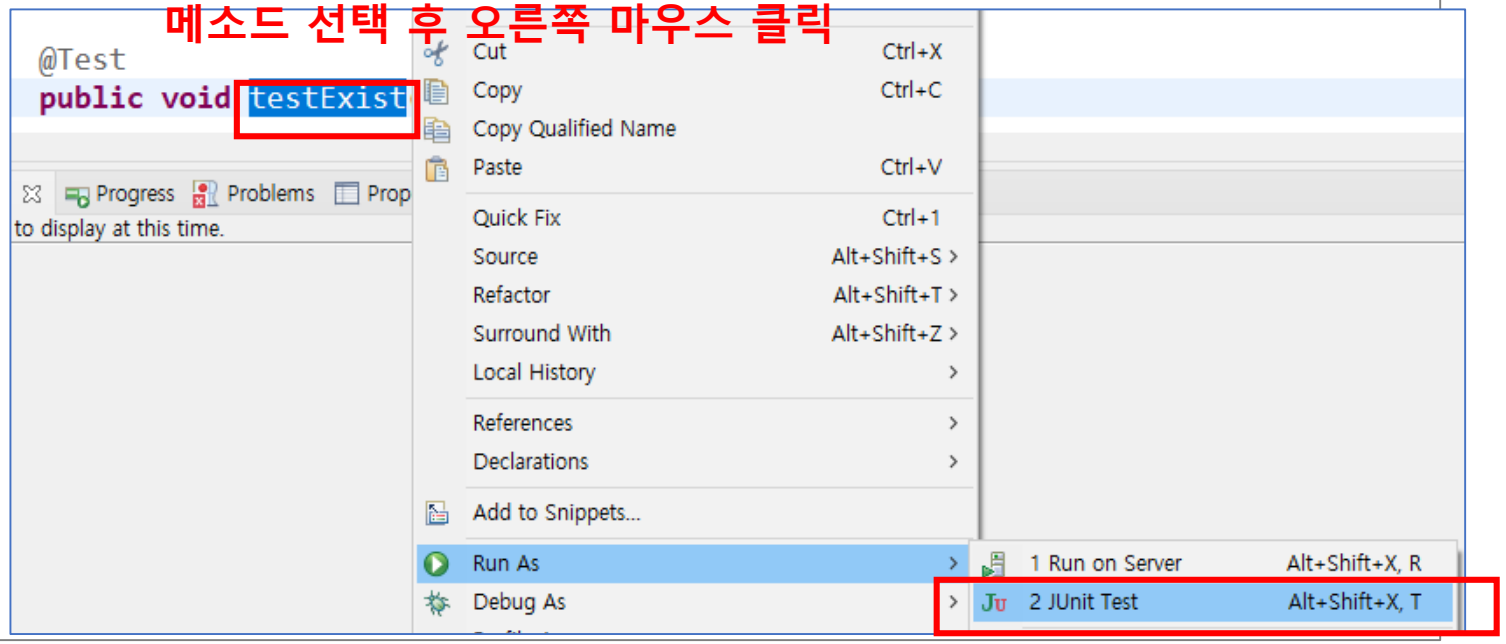
단위 테스트(JUnit) 환경의 구성 및 테스트



아래 세개의 어노테이션은 복사/붙여넣기 할 것

```
@RunWith(SpringJUnit4ClassRunner.class)  //현재 테스트 코드가 스프링 실행 역할을 할 것이라고 알림
@ContextConfiguration("file:src/main/webapp/WEB-INF/spring/root-context.xml") //설정파일 읽어내기
@Log4j //lombok을 이용해 로그를 기록하는 Logger를 변수로 생성
public class SampleTests {

  @Autowired
  private Restaurant restaurant;

  @Test
  public void testExist() {

    log.info(restaurant);
    log.info("--------------------------------");
    log.info(restaurant.getChef());

  }
}
```

메소드 선택 후 오른쪽 마우스 클릭



단위 테스트(JUnit) 환경의 구성 및 테스트

- 테스트의 의미
  1. 테스트 코드가 실행되기 위해 스프링 프레임워크가 동작
  2. 동작 과정에서 필요 객체들이 스프링에 등록됨
  3. 의존성 주입이 필요한 객체는 자동으로 주입이 이루어짐

의존성 주입 방법

1. 필드 주입

```
 8 @Component
 9 @ToString
10 public class Restaurant {
11
12     @Autowired
13     private Chef chef;
14
15 }
```

2. Setter 주입

```
 9 @Component
10 @ToString
11 public class Restaurant {
12
13     @Setter(onMethod_ = {@Autowired})
14     private Chef chef;
15
16 }
```

3. 생성자 주입

```
 8 @Component
 9 @ToString
10 @AllArgsConstructor
11 public class Restaurant {
12
13     private Chef chef;
14
15 }
```

4. Final 필드 자동 주입

```
 5 import lombok.RequiredArgsConstructor;
 6 import lombok.ToString;
 7
 8 @Component
 9 @ToString
10 @RequiredArgsConstructor
11 public class Restaurant {
12
13     private final Chef chef;
14
15 }
```

코드에 사용된 어노테이션

구분	어노테이션	설명
Lombok 관련	@Data	• Setter, getter 메소드 포함 다양한 메소드 생성 • @ToString, @EqualsAndHashCode, @Setter, @Getter 등을 모두 결합한 형태. 세부 설정이 없으면 이 것을 사용
	@Setter, @Getter	Setter, getter 메소드 생성
	@Log4j	로그객체 생성(pom에 설정해야 함)
Spring 관련	@Component	해당 클래스가 스프링에서 관리 대상임을 명시 Component-scan을 통해 조사 하여 @Component 가 있는 클래스를 객체를 생성하여 빈으로 관리
	@Autowired	의존성 주입 명시
Test 관련	@RunWith	테스트 시 필요한 클래스 지정
	@ContextConfiguration	스프링이 실행되면서 읽어들여야 하는 설정 정보 명시
	@Test	해당 메소드가 JUnit상에서 단위 테스트의 대상임을 명시

실습

- cs.dit.sample 패키지에 SampleHotel 객체를 정의
- SampleHotel에서 위에서 정의한 Restaurant 객체를 주입. 아래의 의존성 주입방법을 모두 테스트 해볼것
  - 필드 주입
  - 생성자 주입
  - Setter 주입
  - final 필드 주입
- SampleHotel이 만들어 질 때 Restaurant 객체가 주입되는지 SampleTests객체를 만들어서 테스트 하시오.
- 문제가 발생했을 때 무엇을 확인해야하는지 열거하시오.