콘트롤러 요구분석에 맞춰서 해야할일 지원에 대하여

사용자가 전송한 데이터(폼태그 주로) -> 모델 객체로.
DI.
view로 넘어갈때 어떻게 데이터를 넘기는지.

KB금융그룹 | 국민의 평생 금융파트너

2025년 상반기 K-디지털 트레이닝

스프링 MVC의 기본 구조

[KB] IT's Your Life

KB 국민은행

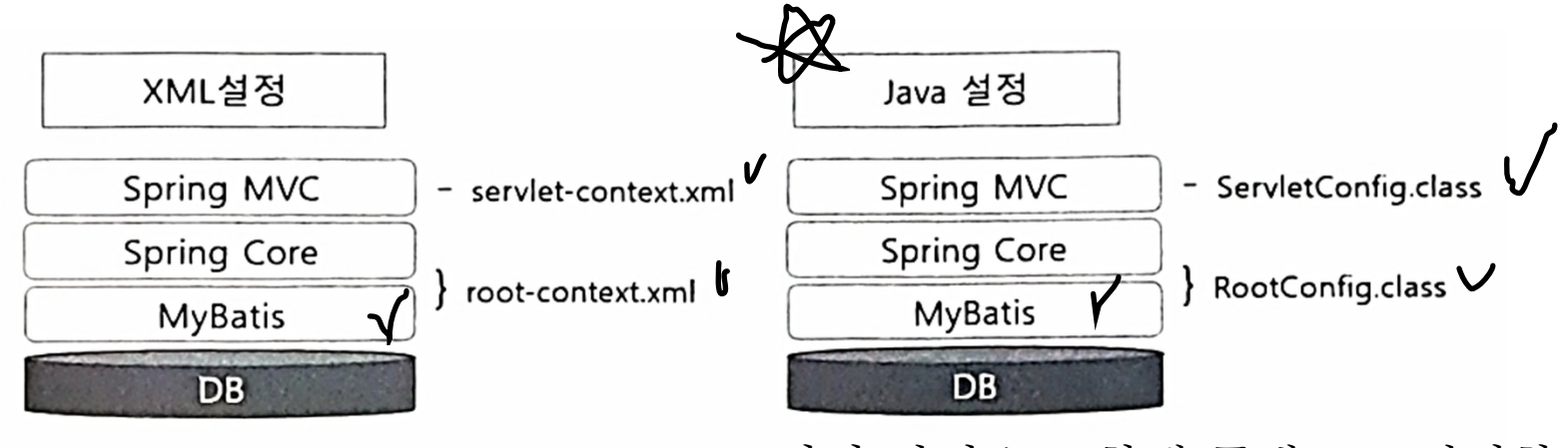
코드가 고정된
프레임워크.

앱설정은
밖으로 빼고.
설정을 어떻게 할것이냐
전통적으로는 xml->복잡함 가독성 떨어짐. 타이핑량 많아.

최근에는 자바 클래스로 설정함.

1 스프링 MVC 프로젝트의 내부 구조

프로젝트 구조



spring core가 DI담당.

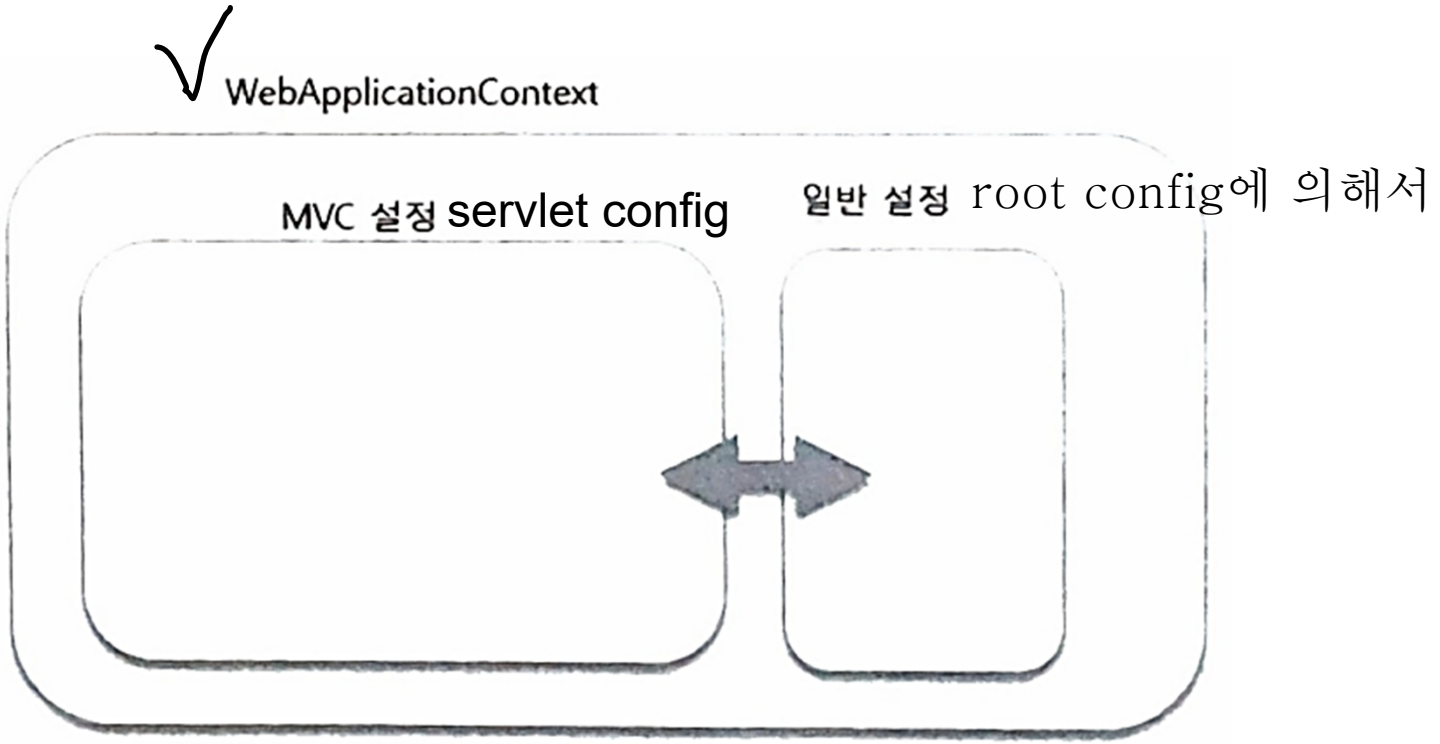
자바 방식으로 할때 클래스로 설정할때

3가지 설정.

root config, servlet config(dispatch servlet)
web config(web.xml대체. 톰캣에게
앱에 대해서 설명. 주로 필터

rootconfig.java 일반설정 범용데이터 관리, 디비 연결 설정.
servletconfig.java frontcontroller설정 컨트롤러 관리
webconfig.java web.xml 설정 앱관련 설정 필터링 캐릭터셋

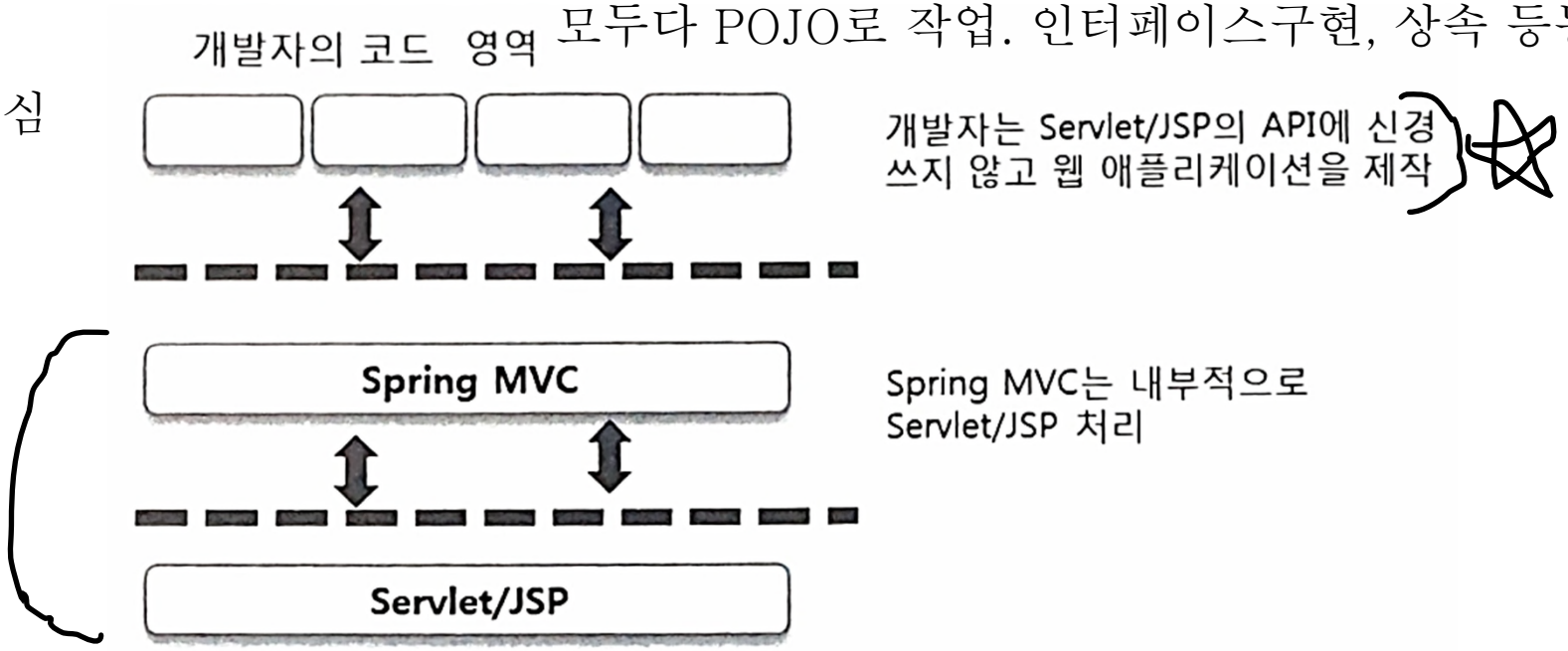
root config에 등록된 빈과 servlet config에 등록된 빈.
후자는 전자를 사용할 수 있음. servletconfig에 등록된 빈은
자신의 의존 객체로 rootconfig에 등록된 빈을 사용할수있음



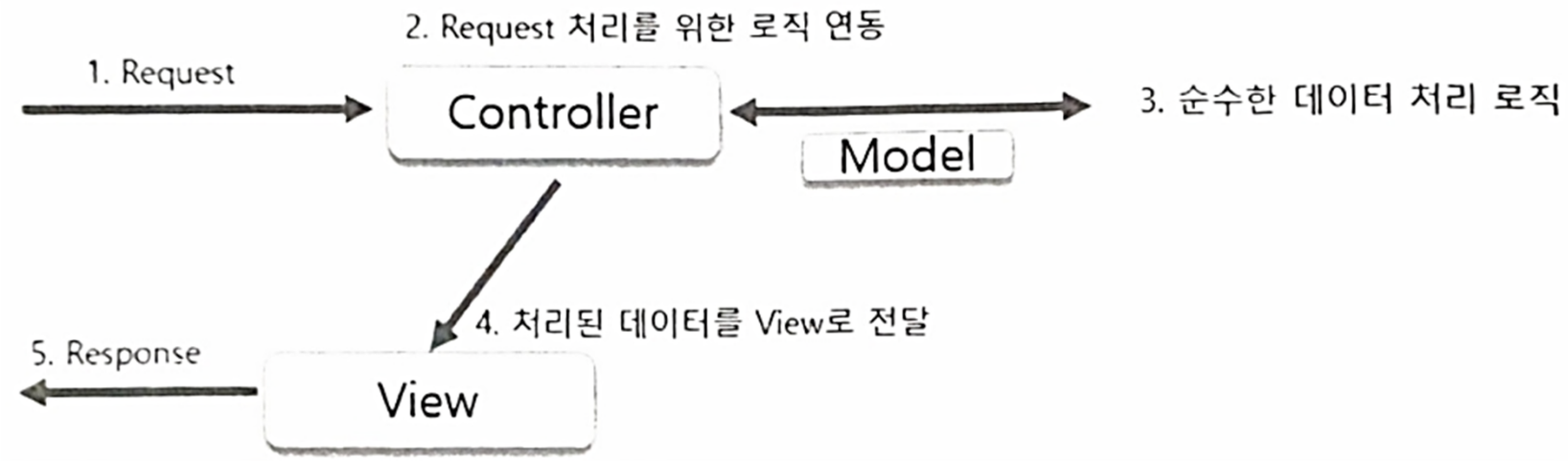
2 스프링 MVC의 기본 사상

모두다 POJO로 작업. 인터페이스구현, 상속 등등 없음

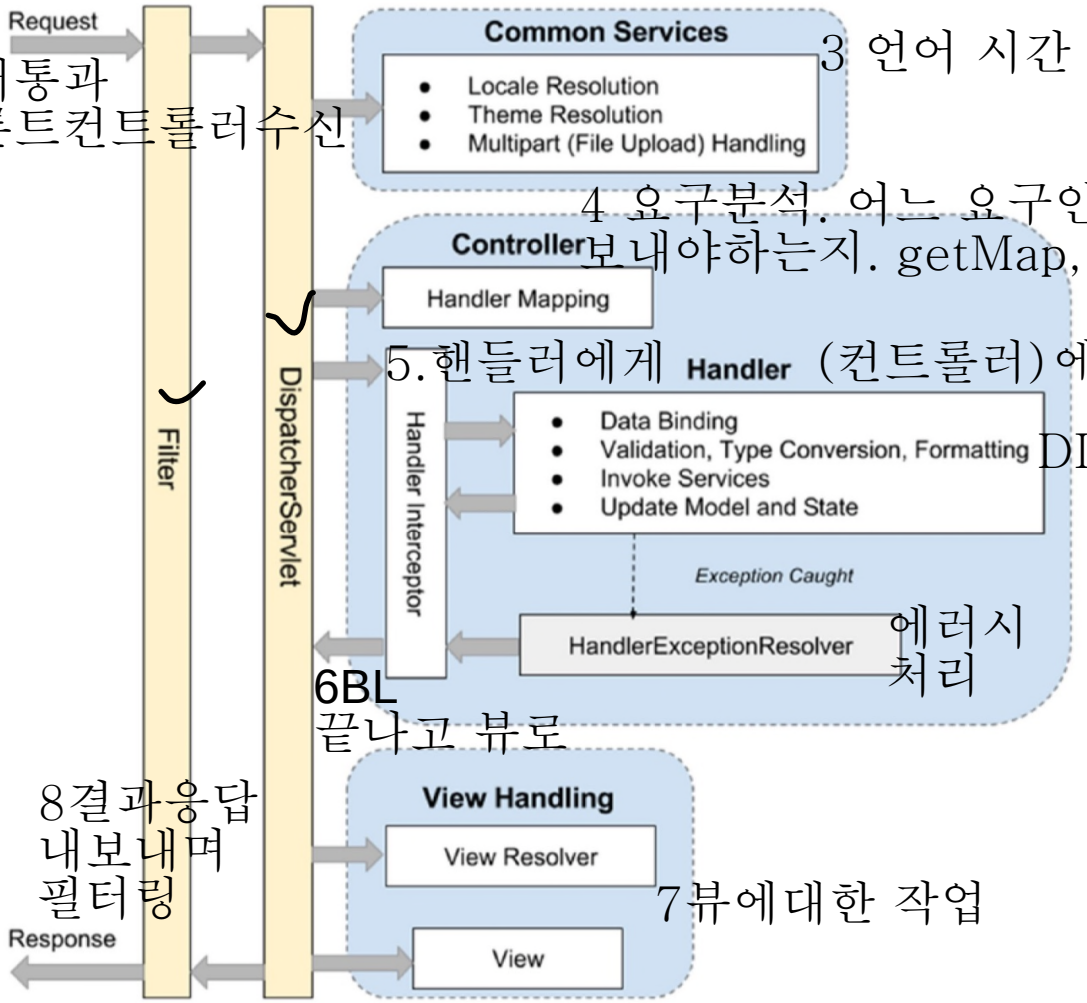
BL중심



모델 2


2. Request 처리를 위한 로직 연동
1. Request
Controller
3. 순수한 데이터 처리 로직
Model
4. 처리된 데이터를 View로 전달
5. Response
View

Spring MVC 라이프 사이클



1필터통과
2프론트컨트롤러수신

3 언어 시간 테마, 파일 처리해석 등

4 요구분석. 어느 요구인지 어떤 컨트롤러로 보내야하는지. getMap, postMap등등

5.핸들러에게 Handler (컨트롤러)에게 넘기기전에 전처리 DI, 데이터, 등등 준비

에러시 처리

6BL 끝나고 뷰로

7뷰에대한 작업

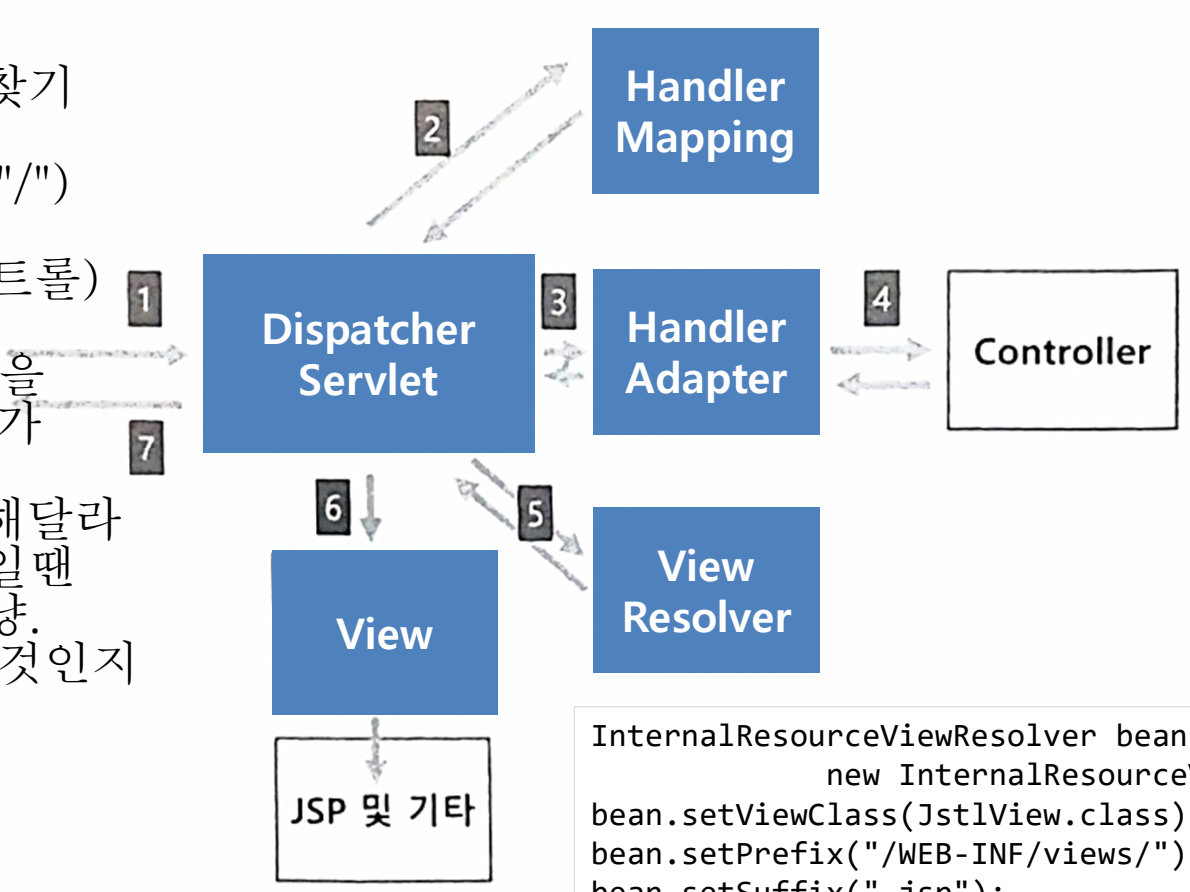
8결과응답 내보내며 필터링

Figure 16-3. *Spring MVC request life cycle*

좀더 단순화한 구조 spring mvc

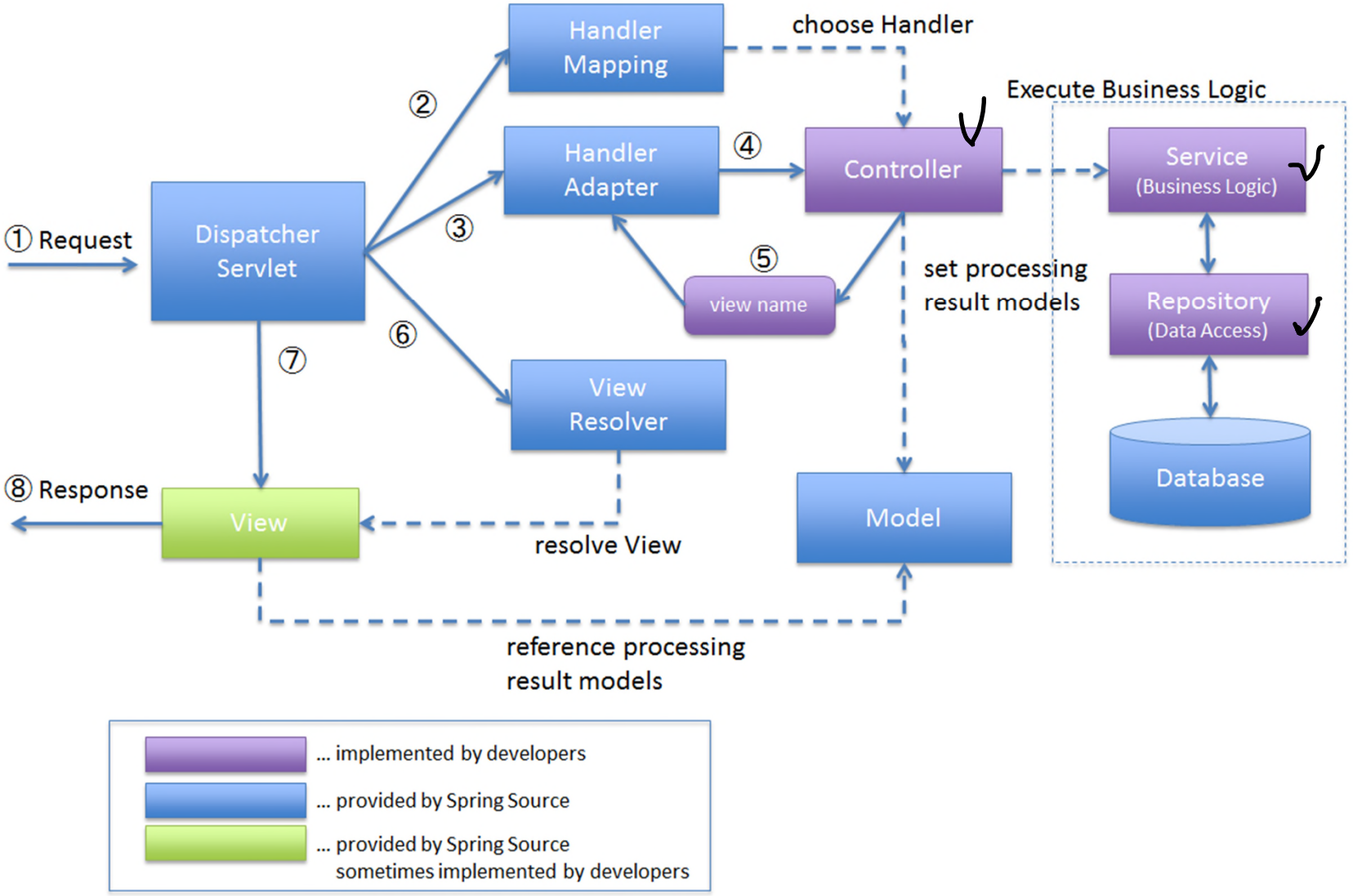
스프링 MVC

1필터통과한 후
요청담당핸들러찾기
2핸들러매핑
@GetMapping("/")
메소드
3찾은핸들러(컨트롤)
에 요청 남김.
4이때필요한D들을
handleradapter가
Injection함.
5뷰의경로 해석해달라
접두어접미어붙일땐
붙이고아닐땐그냥.
어느경로로 넘길것인지
6뷰로.
7출력




```
InternalResourceViewResolver bean =
                new InternalResourceViewResolver();
bean.setViewClass(JstlView.class);
bean.setPrefix("/WEB-INF/views/");
bean.setSuffix(".jsp");
registry.viewResolver(bean);
```

Spring MVC 라이프 사이클



Spring MVC 라이프 사이클

처리 순서

- Filter → DispatcherServlet → HandlerMapping → HandlerInterceptor → Controller → Service → Repository(Mapper) → ViewResolver 순

Filter

- Web Application의 전역적인 로직을 담당
- Filter라는 단어 뜻에서 알 수 있듯이, 전체적인 필터링(설정)을 하는 곳
- DispatcherServlet에 들어가기 전인 Web Application단에서 실행

DispatcherServlet

- 들어오는 모든 Request를 우선적으로 받아 처리해주는 서블릿
- HandlerMapping에게 Request에 대해 매핑할 Controller 검색을 요청
- HandlerMapping으로부터 Controller 정보를 반환받아 해당 Controller와 매핑

→ Request에 대해 어느 컨트롤러로 매핑시킬 것인지 배치하는 역할

Spring MVC 라이프 사이클

- **HandlerMapping**
  - DispatcherServlet으로부터 검색을 요청받은 Controller를 찾아 정보를 리턴

- **HandlerInterceptor**
  - Request가 Controller에 매핑되기 전 앞단에서 부가적인 로직을 추가
  - 주로 세션, 쿠키, 권한 인증 로직에 많이 사용됩니다.

- **Controller**
  - Request와 매핑되는 곳
  - Request에 대해 어떤 로직(Service)으로 처리할 것인지를 결정
  - 그에 맞는 Service를 호출
  - Service Bean을 스프링 컨테이너로부터 주입

Spring MVC 라이프 사이클

- **Service**
  - 데이터 처리 및 가공을 위한 비즈니스 로직을 수행
  - Request에 대한 실질적인 로직을 수행
  - Repository를 통해 DB에 접근하여 데이터의 CRUD(Create, Read, Update, Delete)를 처리

- **Repository (DAO, Data Access Object)**
  - DB에 접근하는 객체. 라고 부릅니다.
  - Service에서 DB에 접근할 수 있게 하여 데이터의 CRUD 처리

- **ViewResolver**
  - Controller에서 리턴한 View의 이름을 DispatcherServlet으로부터 넘겨받고,
  - 해당 View로 forward.

→ 이후, 해당 렌더링된 View 화면은 브라우저로 전송(response)