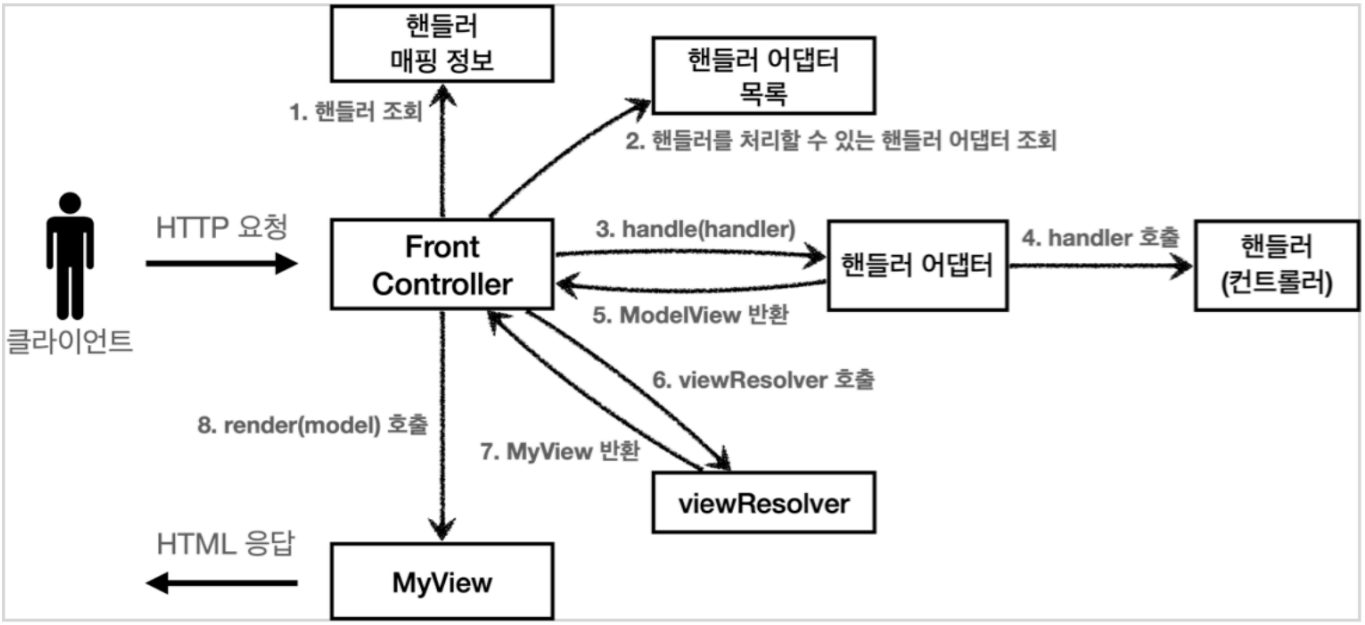
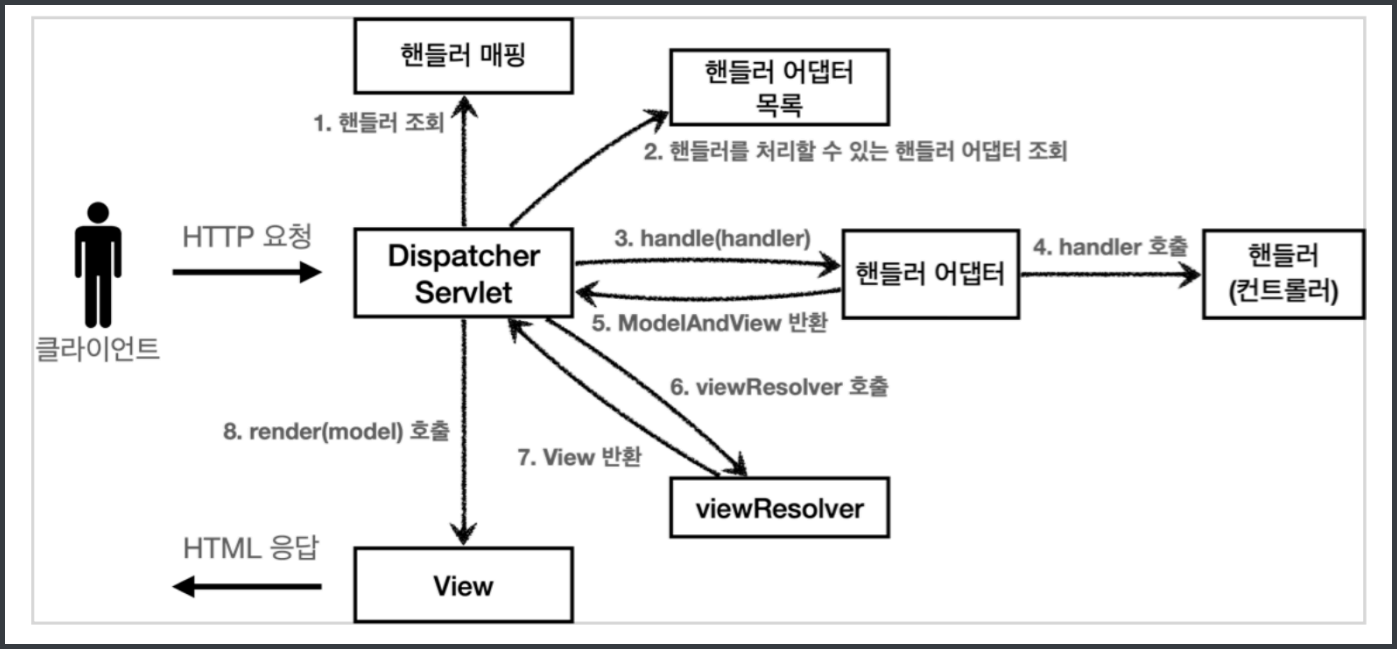
스프링 MVC 전체 구조

직접 만든 MVC 프레임워크 구조





직접 만든 프레임워크 -> 스프링 MVC 비교

- FrontController -> DispatcherServlet
- handlerMappingMap -> HandlerMapping
- ModelView -> ModelAndVeiw
- viewResolver -> ViewResolver

- MyView -> View

DispatcherServlet 구조 살펴보기

`org.springframework,.web.servlet.DsipatcherServlet`

스프링 MVC도 프론트 컨트롤러 패턴으로 구현되어 있다.
스프링 MVC의 프론트 컨트롤러가 바로 디스패처 서블릿(DispatcherServlet)이다.
그리고 이 디스패처 서블릿이 바로 스프링 MVC의 핵심이다.

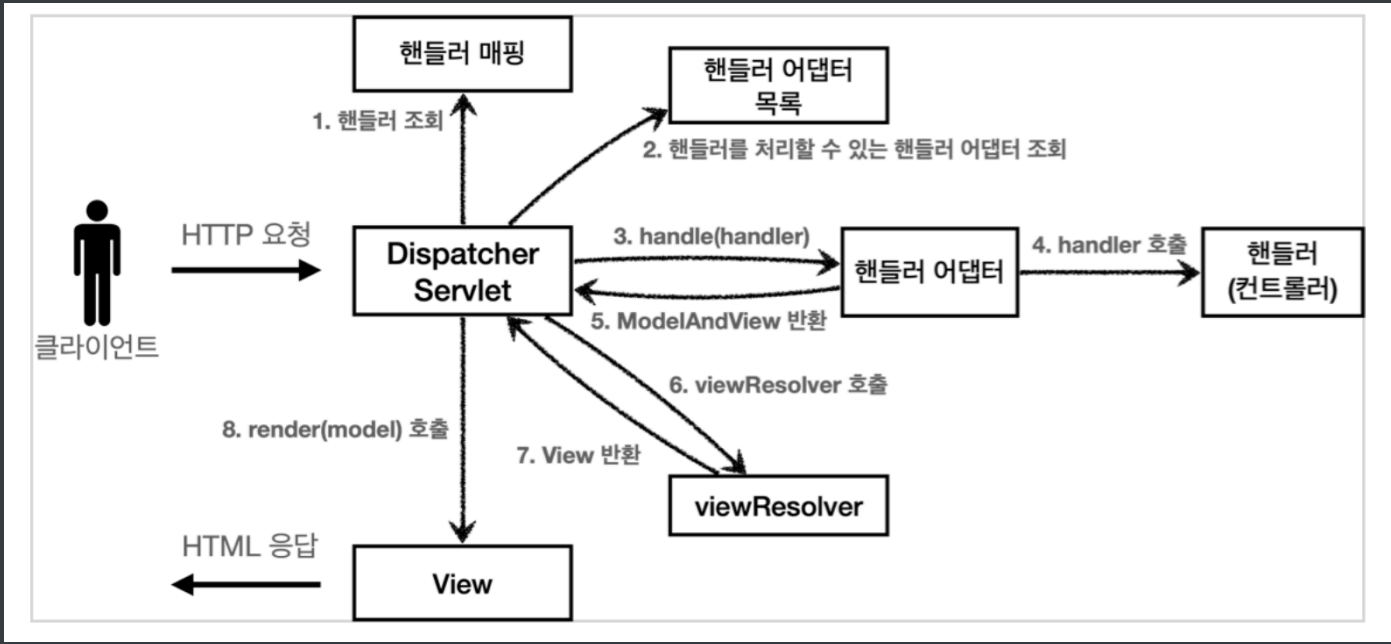
DispatcherServlet 서블릿 등록

- `DispatcherServlet` 도 부모 클래스에서 `HttpServlet` 을 상속받아서 사용하고, 서블릿으로 동작한다.
  - `DispatcherServlet` -> `FrameworkServlet` -> `HttpServletBean` -> `HttpServlet`
- 스프링 부는 `DispatcherServlet` 을 서블릿으로 자동으로 등록하면서 모든경로 (`urlPattern="/"`)에 대해서 매핑한다.
  - 참고 : 더 자세한 경로가 우선순위가 높다. 그래서 기존에 등록한 서블릿도 함께 동작한다.

요청 흐름

- 서블릿이 호출되면 `HttpServlet` 이 제공하는 `service()` 가 호출된다.
- 스프링 MVC는 `DispatcherServlet` 의 부모인 `FrameworkServlet` 에서 `service()` 를 오버라이드 해두었다.
- `FrameworkServlet.service()` 를 시작으로 여러 메서드가 호출되면서 `DispatcherServlet.doDispatch()` 가 호출된다.

SpringMVC 구조



동작 순서

1. **핸들러 조회** : 핸들러 매핑을 통해 URL에 매핑된 핸들러(컨트롤러)를 조회한다.
2. **핸들러 어댑터 조회** : 핸들러를 실행할 수 있는 핸들러 어댑터를 조회한다.
3. **핸들러 어댑터 실행** : 핸들러 어댑터를 실행한다.
4. **핸들러 실행** : 핸들러 어댑터가 실제 핸들러를 실행한다.
5. **ModelAndView 반환** : 핸들러 어댑터는 핸들러가 반환하는 정보를 ModelAndView로 변환 해서 반환한다.
6. **viewResolver 호출** : 뷰 리졸버를 찾고 실행한다.
    - JSP의 경우 : `InternalResourceViewResolver` 가 자동으로 등록되고, 사용된다.
7. **View 반환** : 뷰 리졸버는 뷰의 논리 이름을 물리 이름으로 바꾸고, 렌더링 역할을 담당하는 뷰 객체를 반환한다.
8. **뷰 렌더링** : 뷰를 통해서 뷰를 렌더링 한다.

인터페이스 살펴보기

- 스프링 MVC의 큰 장점은 `DispatcherServelt` 코드의 변경 없이, 원하는 기능을 변경하거나 확장할 수 있다는 점이다.
  지금까지 설명한 대부분을 확장 가능할 수 있게 인터페이스로 제공한다.
- 이 인터페이스들만 구현해서 `DispatcherServlet` 에 등록하면 여러분만의 컨트롤러를 만들 수 있다.

주요 인터페이스 목록

- 핸들러 매핑 : `org.springframework.web.servlet.HandlerMapping`
- 핸들러 어댑터 : `org.springframework.web.servlet.HandlerAdapter`
- 뷰 리졸버 : `org.springframework.web.servlet.ViewResolver`
- 뷰 : `org.springframework.web.servlet.View`

정리

스프링 MVC는 코드 분량도 매우 많고, 복잡해서 내부 구조를 다 파악하는 것은 쉽지 않다. 사실 해당 기능을 직접 확장하거나 나만의 컨트롤러를 만드는 일은 없으므로 걱정하지 않아도 된다,. 왜냐면 스프링 MVC는 전세계 수 많은 개발자들의 요구사항에 맞추어 기능을 계속 확장해왔고, 그래서 여러분이 웹 애플리케이션을 만들 때 필요로 하는 부분의 기능이 이미 다 구현되어 있다.
그래도 이렇게 핵심 동작방식을 알아두어야 향후 문제가 발생했을 때 어떤 부분에서 문제가 발생했는지 쉽게 파악하고, 문제를 해결할 수 있다. 그리고 확장 포인트가 필요할 때, 어떤 부분을 확장해야 할 지 감을 잡을 수 있다. 실제 다른 컴포넌트를 제공하거나 기능을 확장하는 부분들은 강의를 진행하면서 조금씩 설명하겠다. 지금은 전체적인 구조가 이렇게 되어 있구나 하고 이해하면 된다.

우리가 지금까지 함계 개발한 MVC 프레임워크와 유사한 구조여서 이해하기 어렵지 않았을 것이다.