[Spring] MVC 1편 - 05. 스프링 MVC - 구조 이해

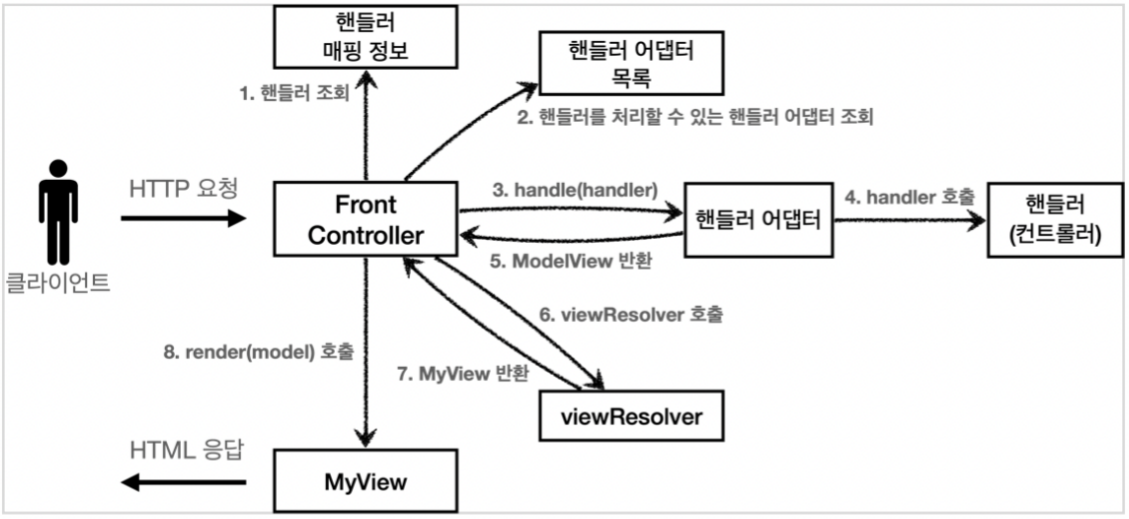
> 이 글은 스프링 [스프링 MVC 1편]을 듣고 정리한 내용입니다

> 앞으론 정리 주요내용만 간단하게 하도록 하자 꼭 ....

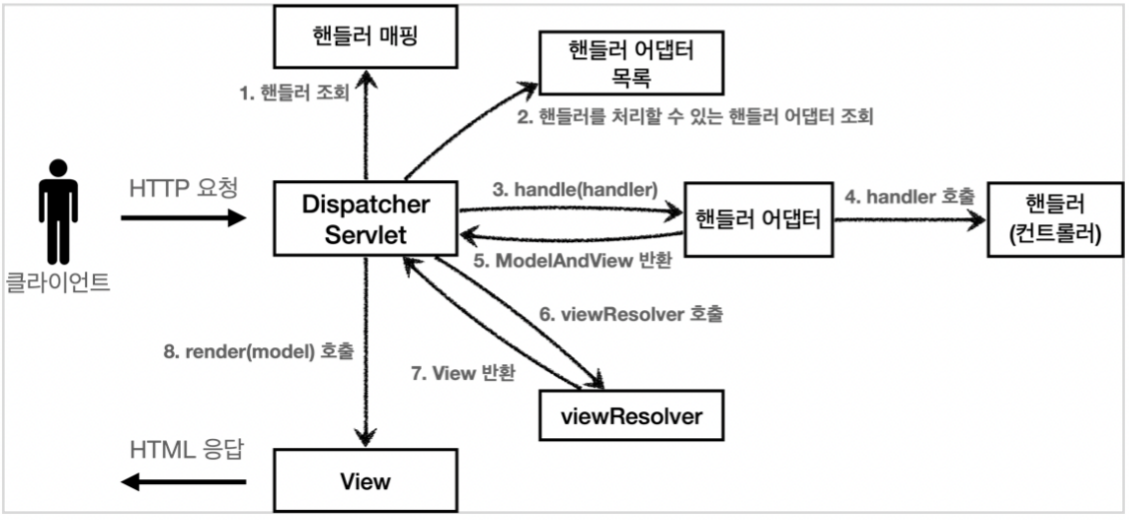
📌 스프링 MVC 전체 구조

- 저번시간까지 직접 만든 MVC 프레임워크와 스프링 MVC를 비교해보자

직접 만든 MVC 프레임워크 구조



SpringMVC 구조



DispatcherServlet 구조

`org.springframework.web.servlet.DispatcherServlet`

- 스프링 MVC도 프론트 컨트롤러 패턴으로 구현되어 있다.
- 스프링 MVC의 프론트 컨트롤러가 **디스패쳐 서블릿(DispatcherServlet)이다 -> 이것이 바로 스프링 MVC의 핵심이다.

DispatcherServlet 서블릿 등록

- `DispatcherServlet`도 부모 클래스에서 `HttpServlet`을 상속받아서 사용하고, 서블릿으로 동작한다.
- 스프링 부트는 `DispatcherServlet`을 서블릿으로 자동 등록하면서 **모든 경로*** (`urlPatterns="/"`)에 대해서 매핑한다

요청 흐름

- 서블릿이 호출되면 `HttpServlet`이 제공하는`service()`가 호출된다.

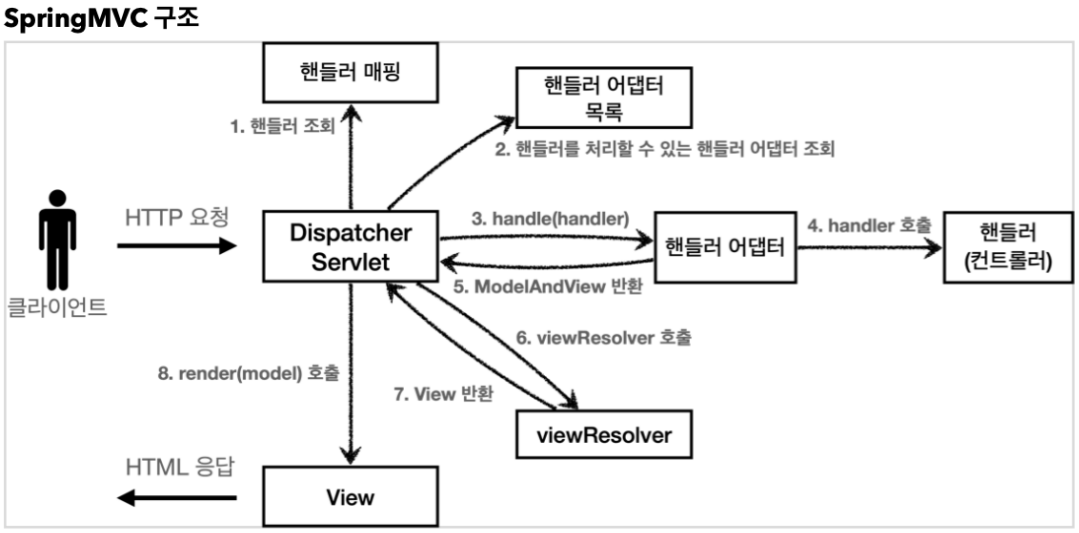
DispatcherServlet.doDispatch()

```
protected void doDispatch(HttpServletRequest request, HttpS
ervletResponse
            response) throws Exception {
        HttpServletRequest processedRequest = request;
```

```
        HandlerExecutionChain mappedHandler = null;
        ModelAndView mv = null;

        // 1. 핸들러 조회
        mappedHandler = getHandler(processedRequest);
        if (mappedHandler == null) {
            noHandlerFound(processedRequest, response);
            return;
        }

        //2.핸들러 어댑터 조회-핸들러를 처리할 수 있는 어댑터
        HandlerAdapter ha = getHandlerAdapter(mappedHandle
r.getHandler());

        // 3. 핸들러 어댑터 실행 -> 4. 핸들러 어댑터를 통해 핸들러
실행 -> 5. ModelAndView 반환
        mv = ha.handle(processedRequest, response, mappedHa
ndler.getHandler());
        processDispatchResult(processedRequest, response, m
appedHandler, mv,
                dispatchException);
    }

    private void processDispatchResult(HttpServletRequest r
equest, HttpServletResponse response, HandlerExecutionChain
mappedHandler, ModelAndView mv, Exception exception) throws
Exception {
        // 뷰 렌더링 호출
        render(mv, request, response);
    }

    protected void render(ModelAndView mv, HttpServletReque
st request, HttpServletResponse response) throws Exception
{
        View view;
        String viewName = mv.getViewName(); //6. 뷰 리졸버를
통해서 뷰 찾기,7.View 반환
        view = resolveViewName(viewName, mv.getModelInterna
```

```
l(), locale, request);

        // 8. 뷰 렌더링
        view.render(mv.getModelInternal(), request, respons
e);
    }
```

- 이 구조와 동작순서를 알고 있자



동작 순서

1. 핸들러 조회: 핸들러 맵이을 통해 요청 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. ModelView 반환: 핸들러 어댑터는 핸들러가 반환하는 정보를 ModelAndView로 변환해서 반환한다.
6. viewResolver 호출: 뷰 리졸버를 찾고 실행한다.
    - JSP의 경우, `InternalResourceViewResolver`가 자동 등록되고, 사용된다.
7. View 반환: 뷰 리졸버는 뷰의 논리 이름을 물리 이름으로 바꾸고, 렌더링 역할을 담당하는 뷰 객체를 반환한다.

- JSP의 겨우 `InternalResourceView(JstlView)`를 반환하는데, 내부에 `forward()` 로직이 있다.

8. 뷰 렌더링: 뷰를 통해서 뷰를 렌더링한다.