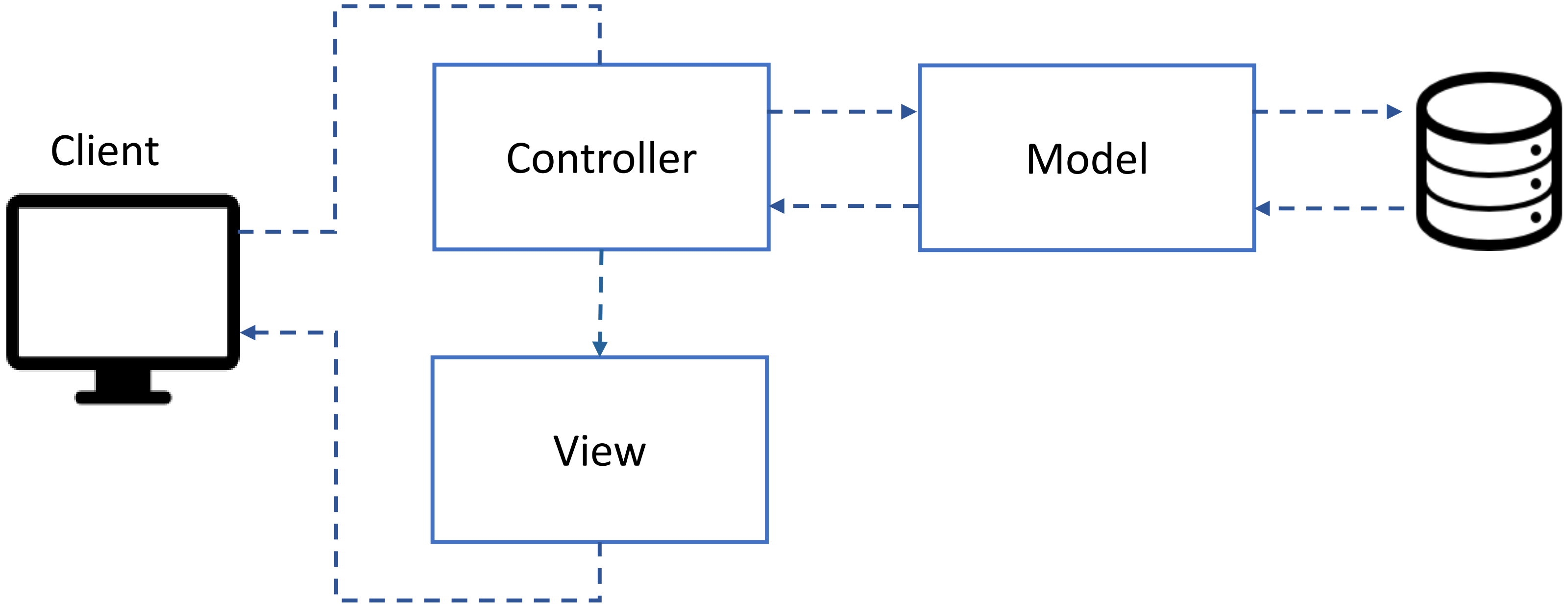
SPRING MVC & REST API

MVC

Model – View – Controller

MVC

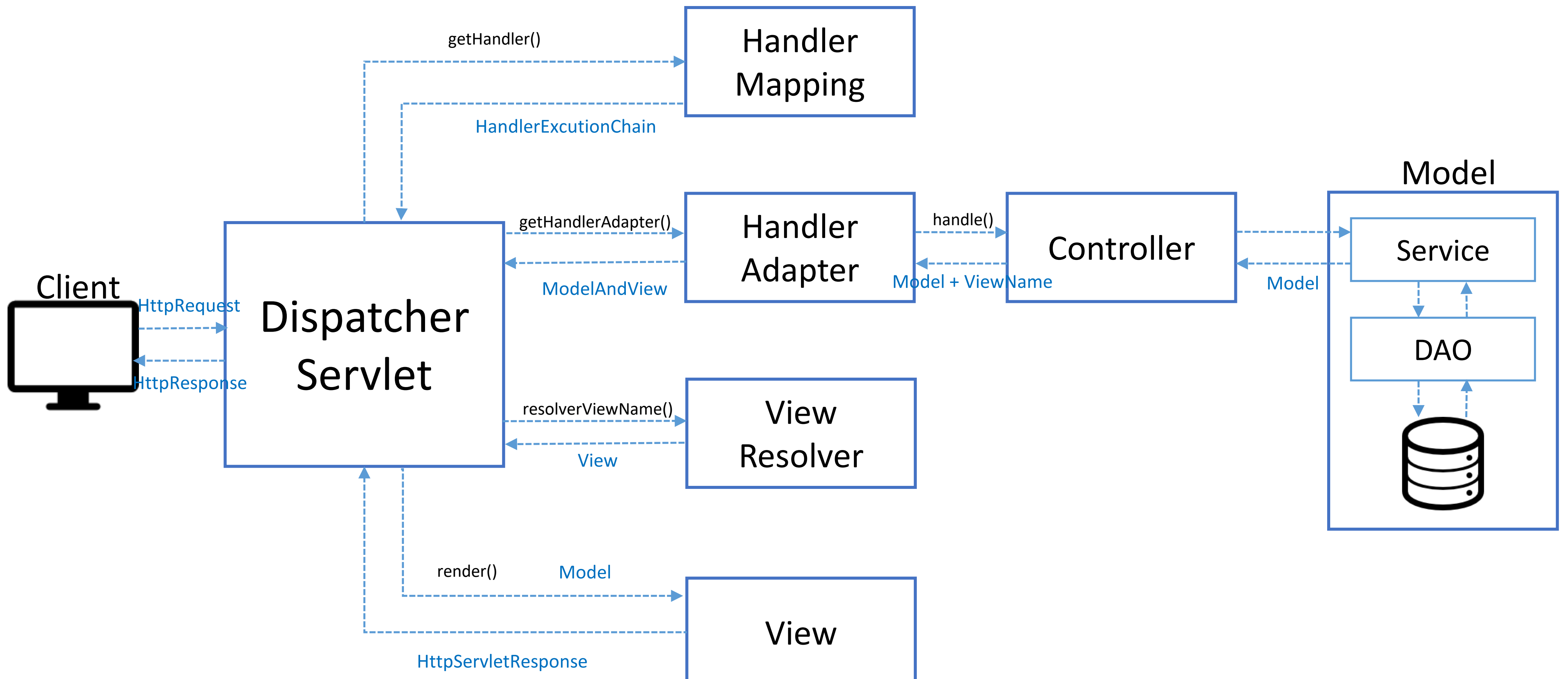
: 클라이언트의 요청처리와 응답처리, 비즈니스 로직 처리하는 부분을 모듈화시킨 구조.



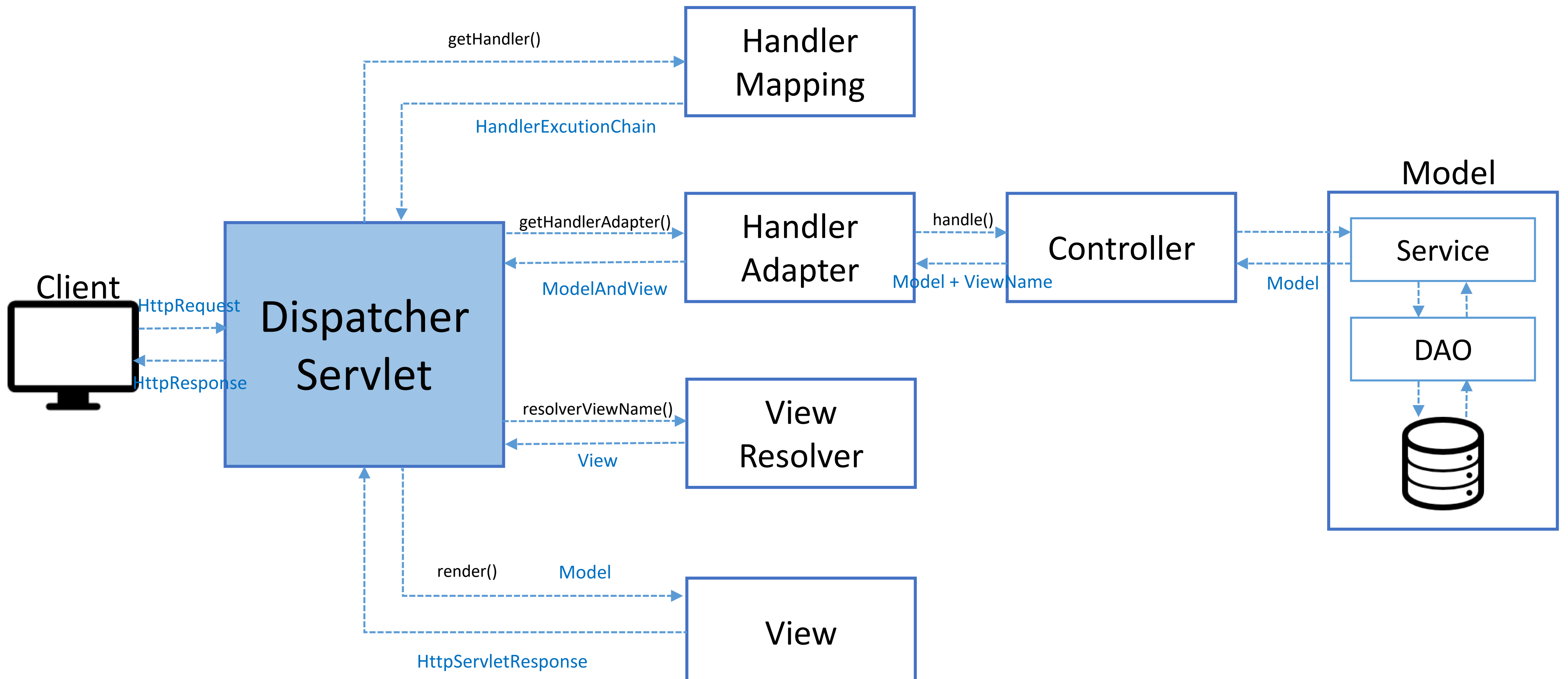
- 장점 : 처리작업의 분리로 인해 유지보수와 확장이 용이하다.
- 단점 : 구조 설계를 위한 시간이 많이 소요되므로 개발 기간이 증가한다.

?

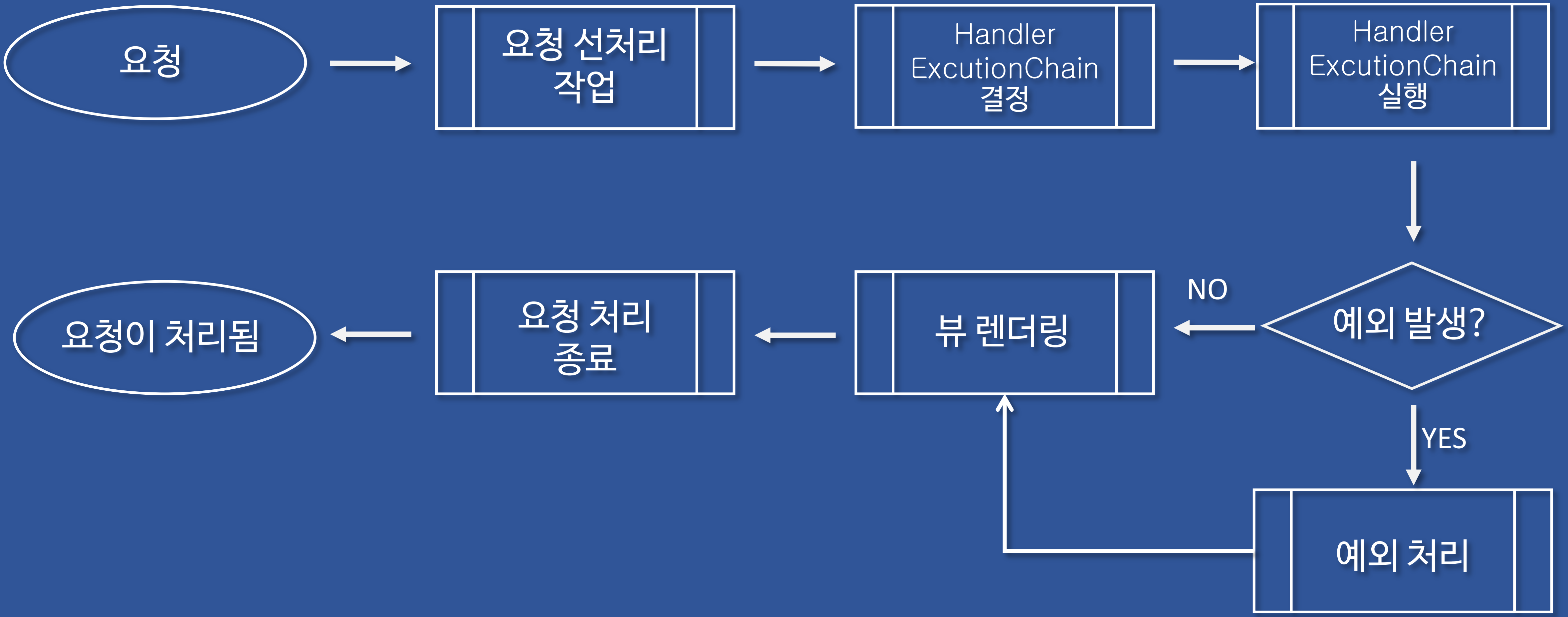
스프링 MVC 요청 처리 과정



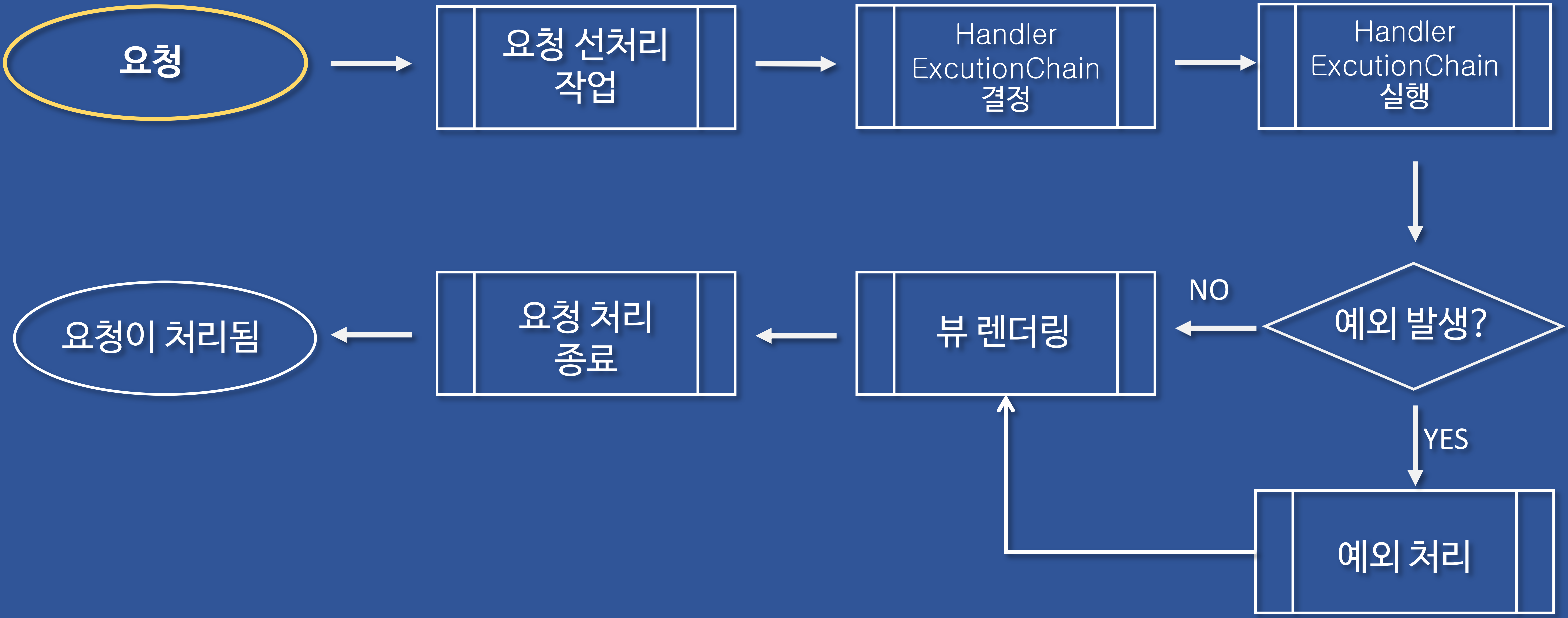
스프링 MVC 요청 처리 과정



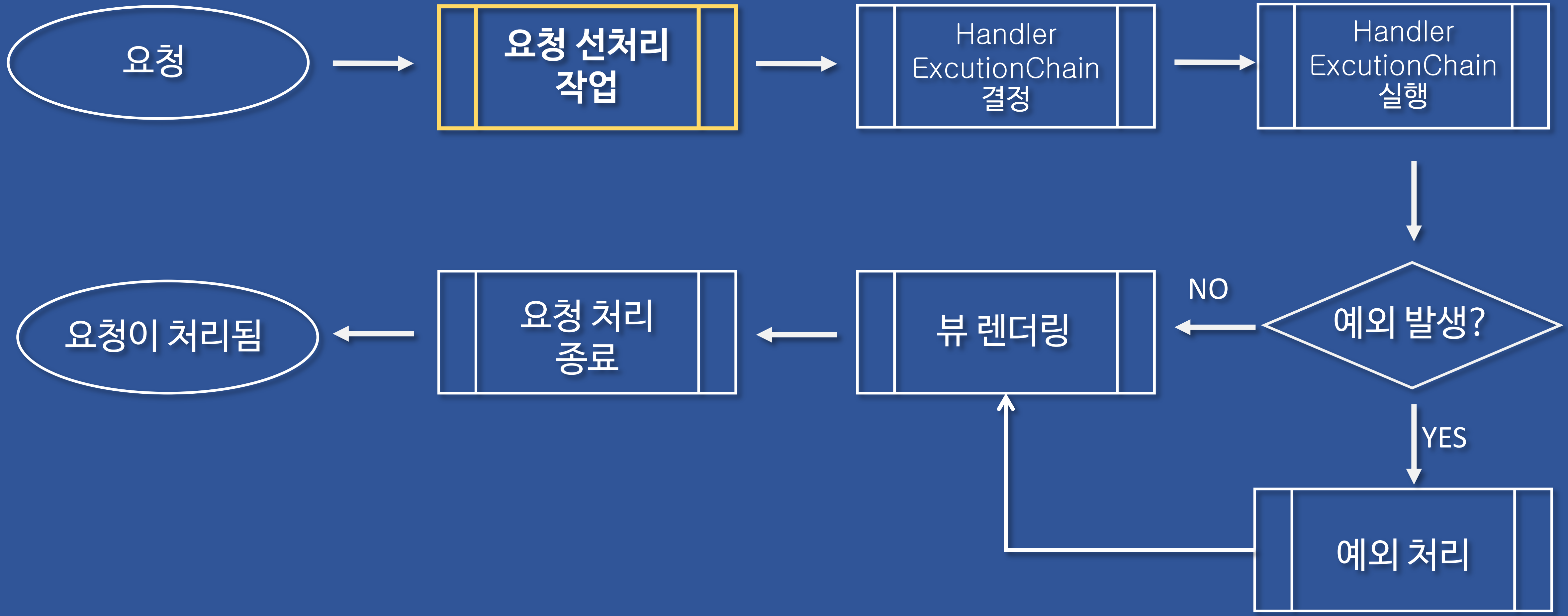
Dispatcher Servlet 내부 요청 처리과정



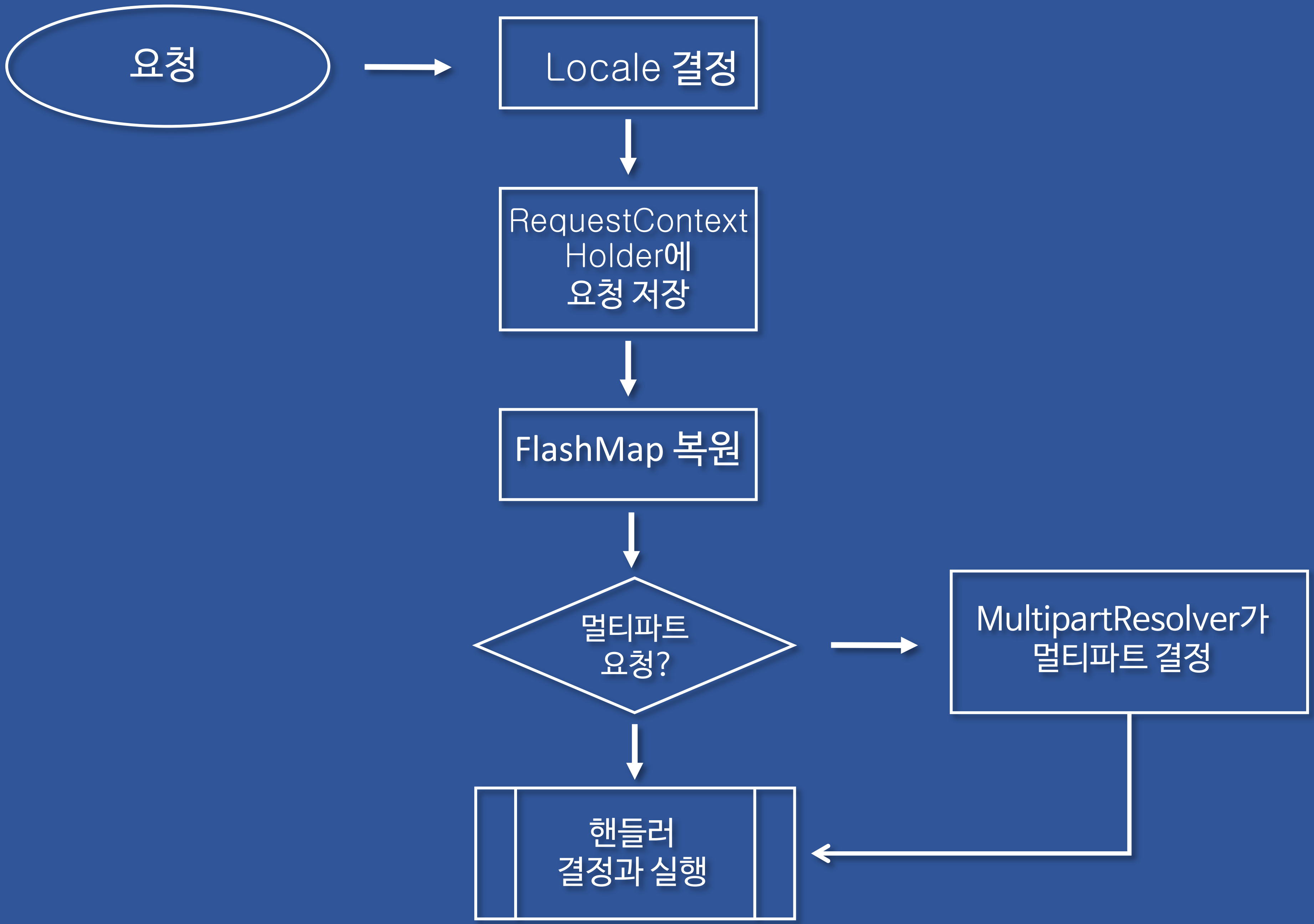
Dispatcher Servlet 내부 요청 처리과정



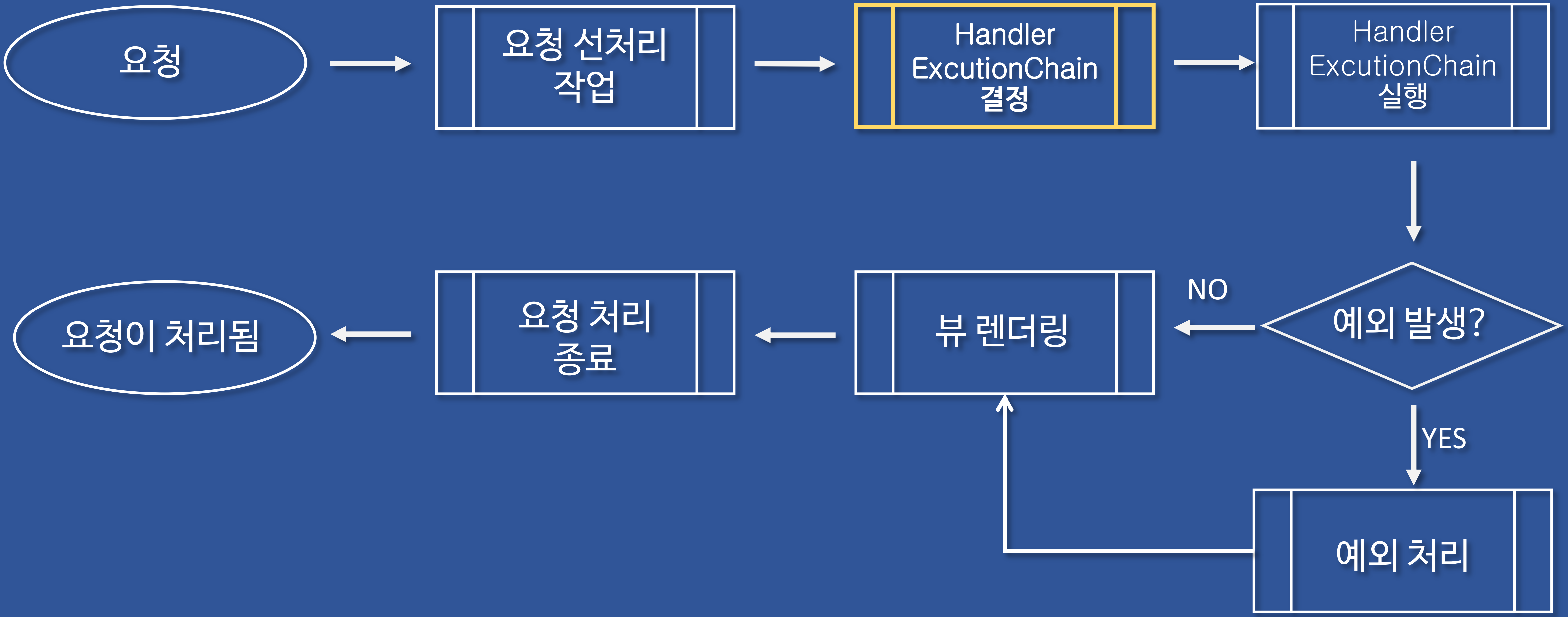
Dispatcher Servlet 내부 요청 처리과정



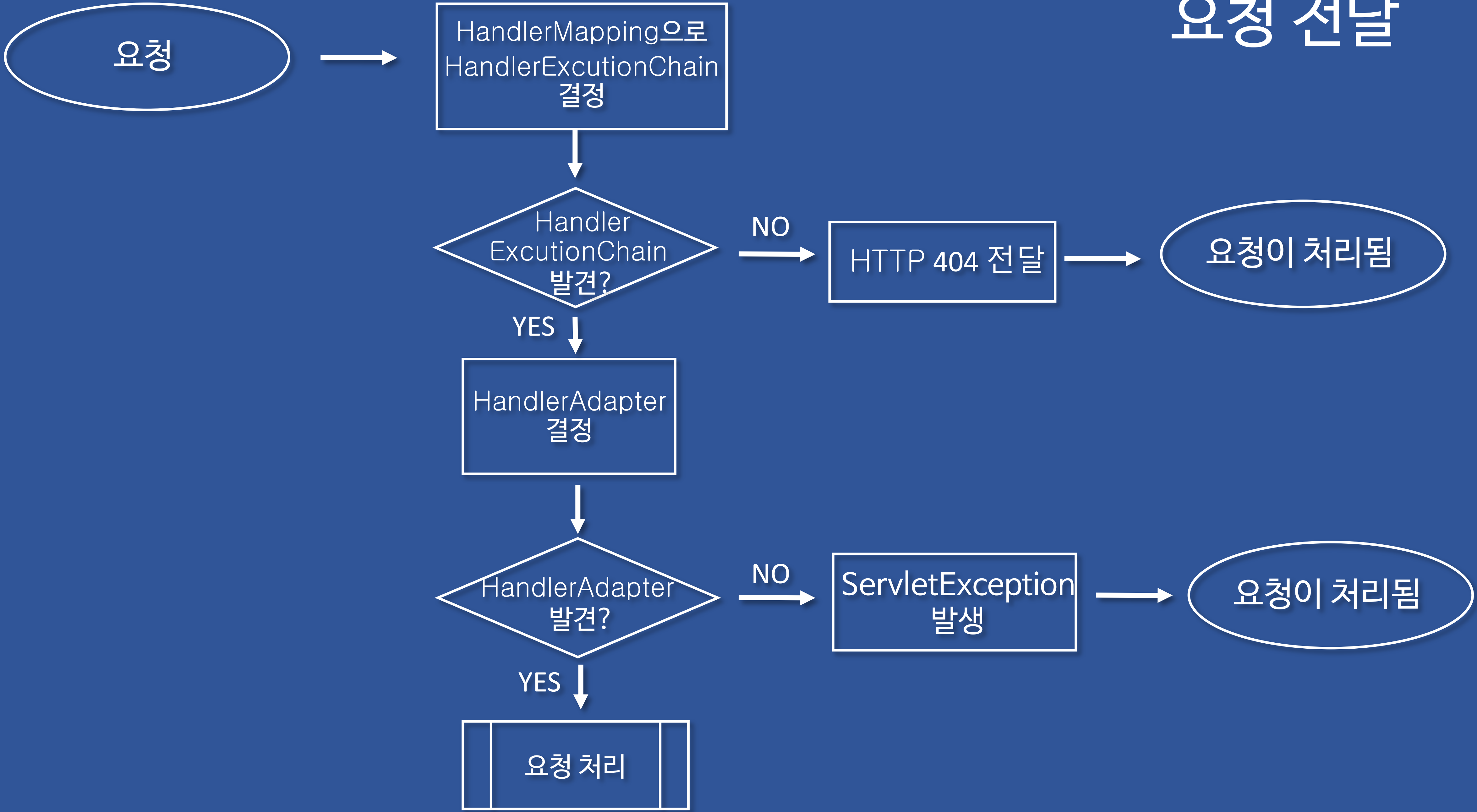
선처리 작업


요청
Locale 결정
RequestContext Holder에 요청 저장
FlashMap 복원
멀티파트 요청?
MultipartResolver가 멀티파트 결정
핸들러 결정과 실행

Dispatcher Servlet 내부 요청 처리과정

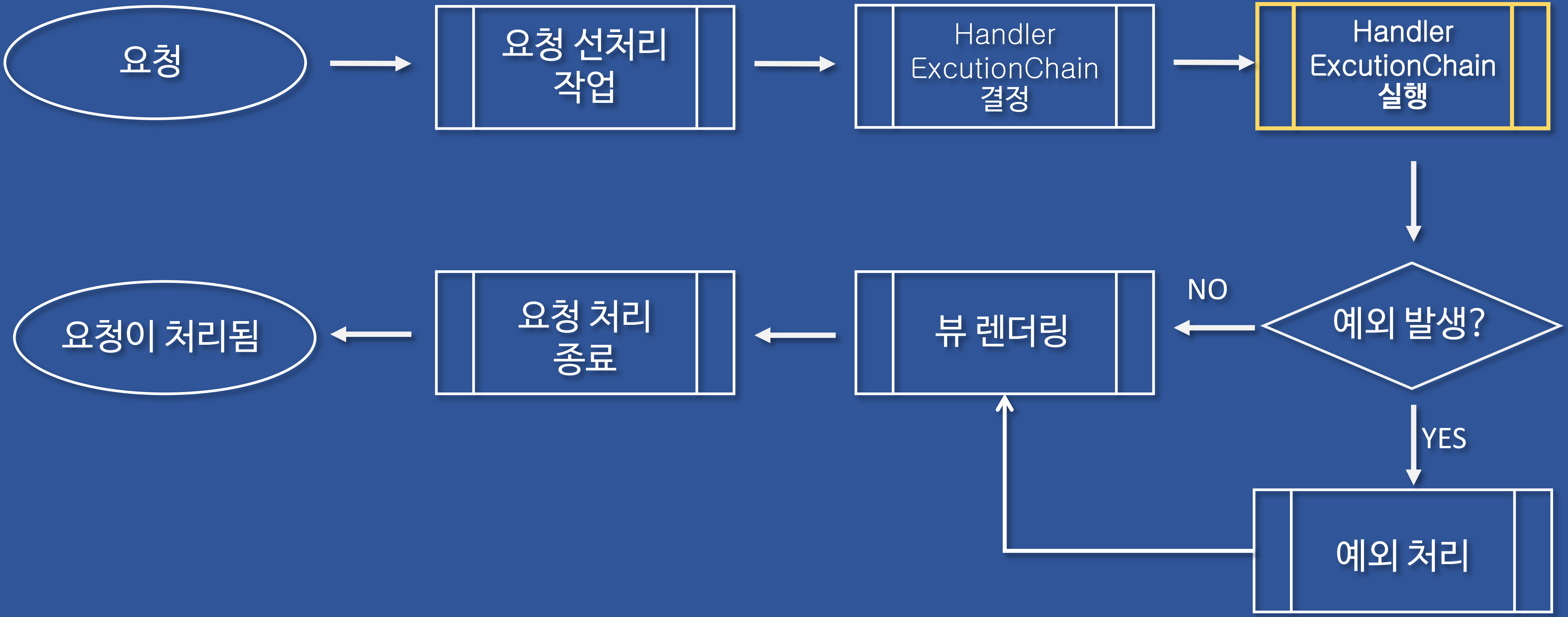


요청 전달

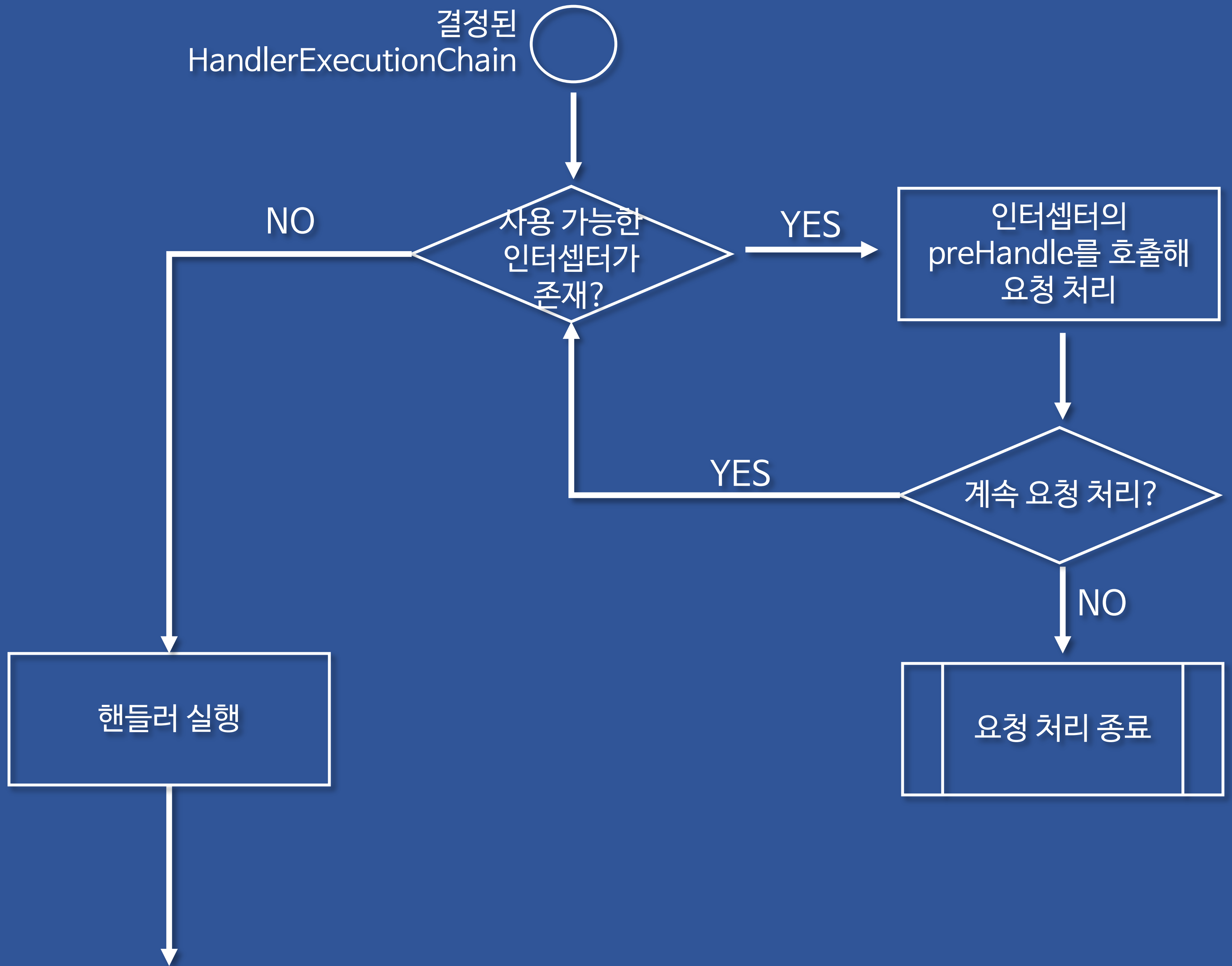


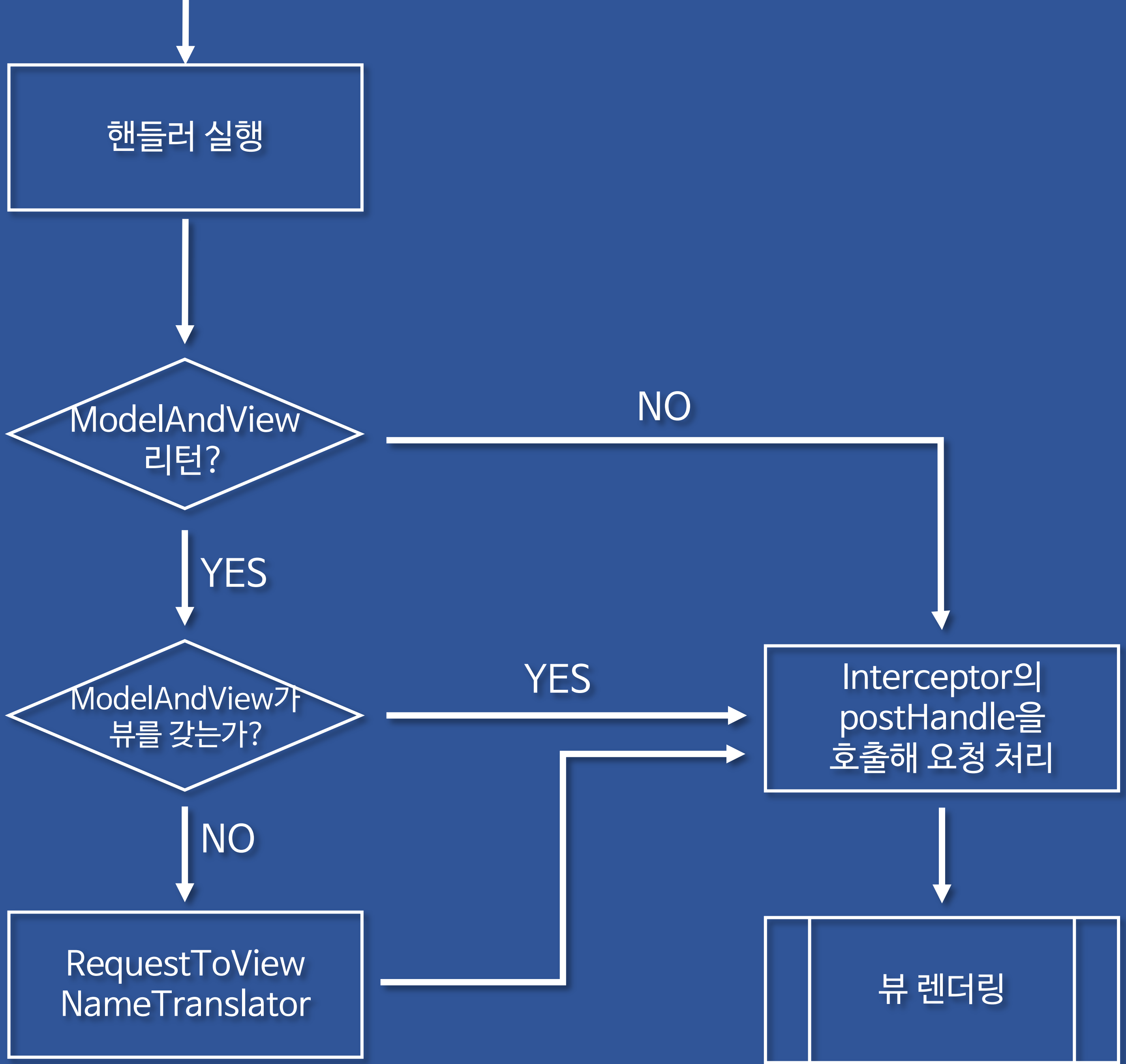
요청
HandlerMapping으로 HandlerExcutionChain 결정
Handler ExcutionChain 발견?
NO
HTTP 404 전달
요청이 처리됨
YES
HandlerAdapter 결정
HandlerAdapter 발견?
NO
ServletException 발생
요청이 처리됨
YES
요청 처리

Dispatcher Servlet 내부 요청 처리과정

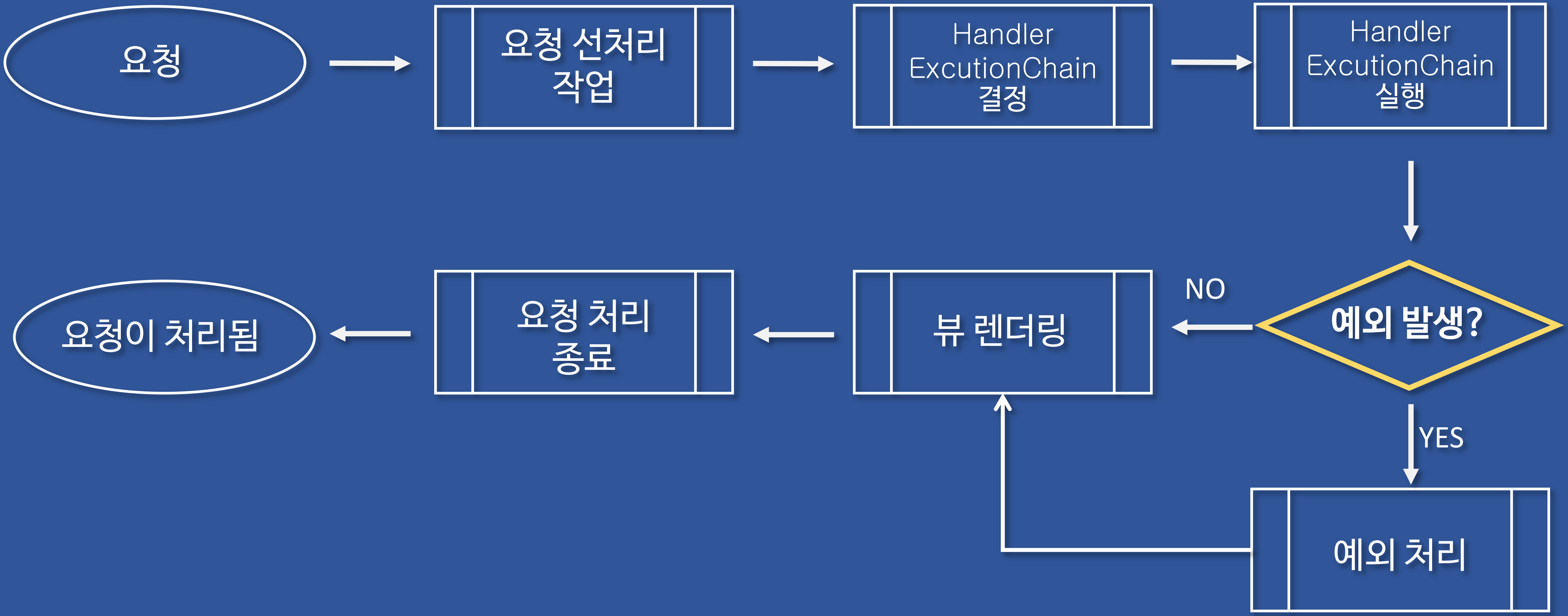


요청 처리

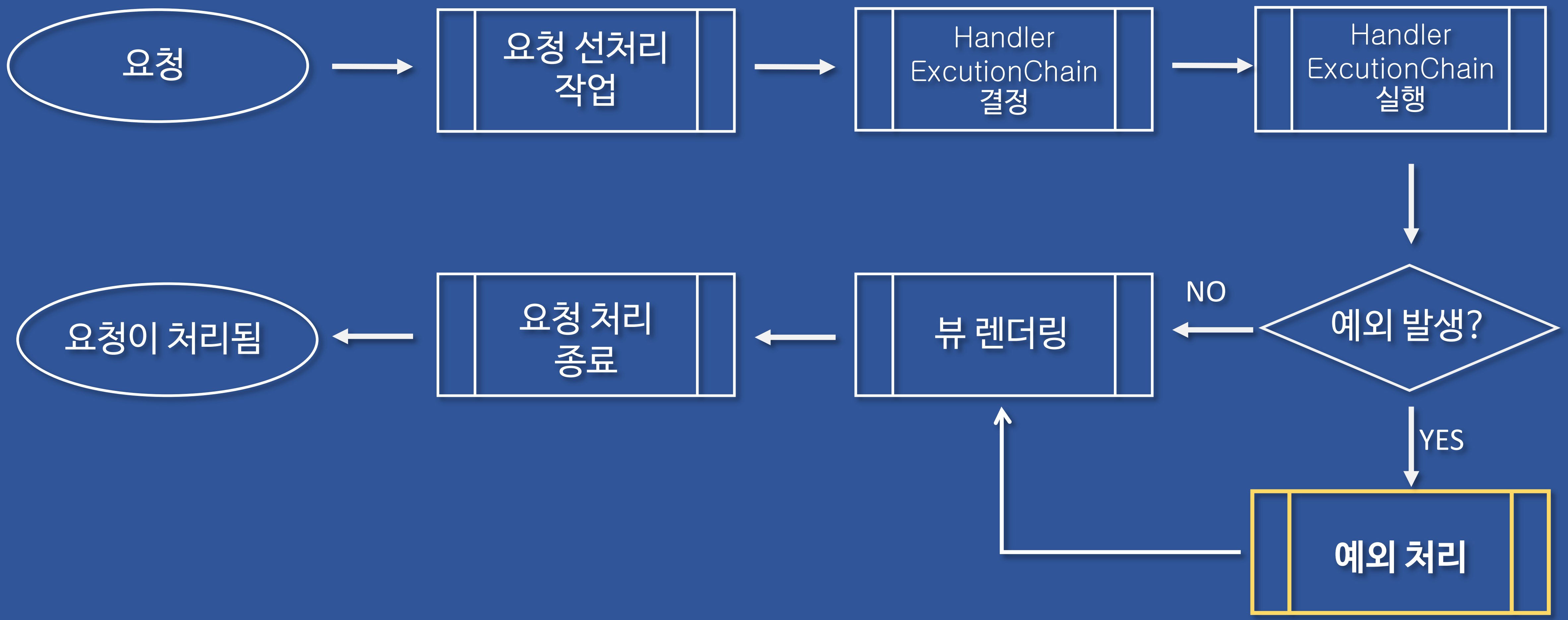

결정된 HandlerExecutionChain
사용 가능한 인터셉터가 존재?
NO
YES
인터셉터의 preHandle를 호출해 요청 처리
계속 요청 처리?
YES
NO
핸들러 실행
요청 처리 종료


핸들러 실행
ModelAndView 리턴?
NO
YES
ModelAndView가 뷰를 갖는가?
YES
NO
RequestToView NameTranslator
Interceptor의 postHandle을 호출해 요청 처리
뷰 렌더링

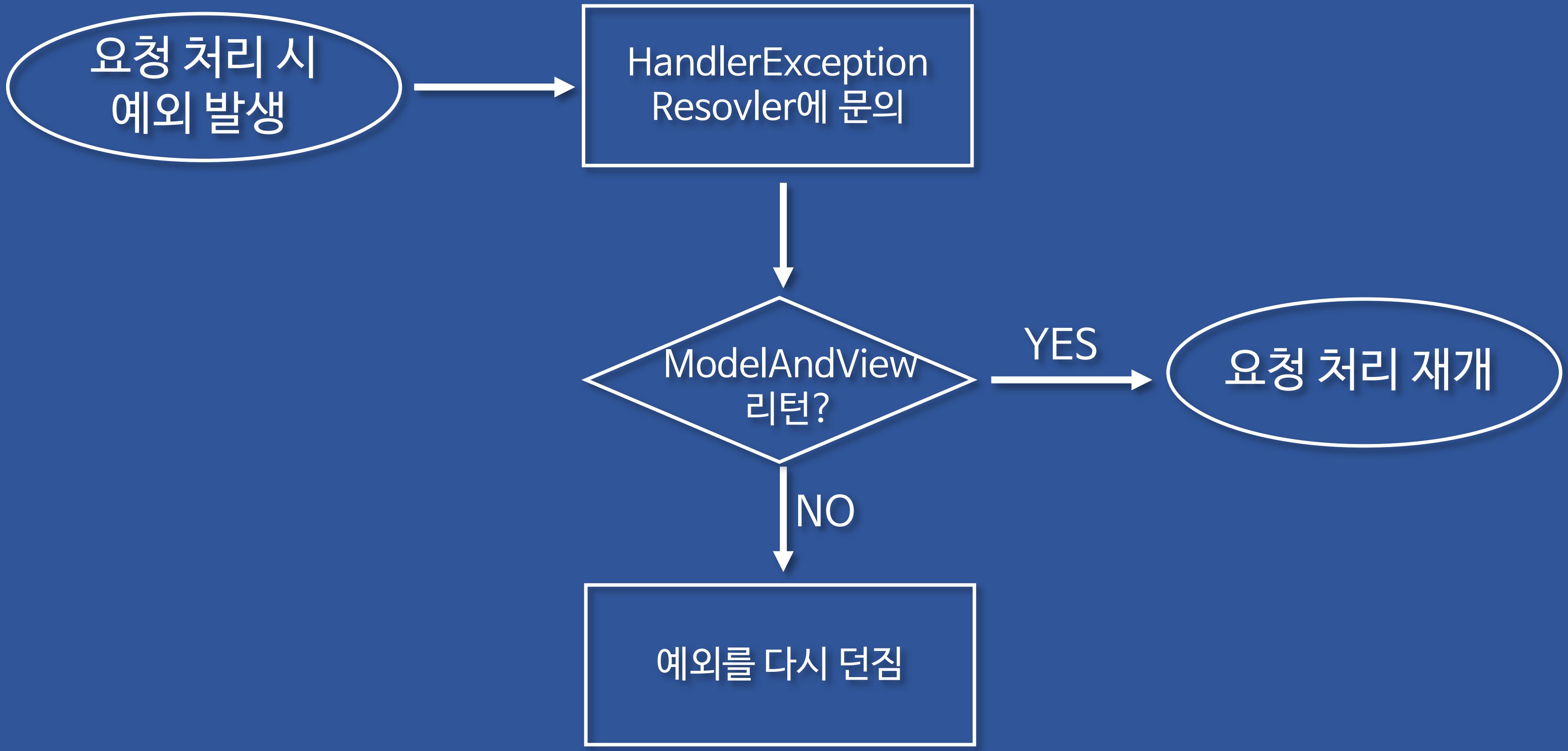
Dispatcher Servlet 내부 요청 처리과정



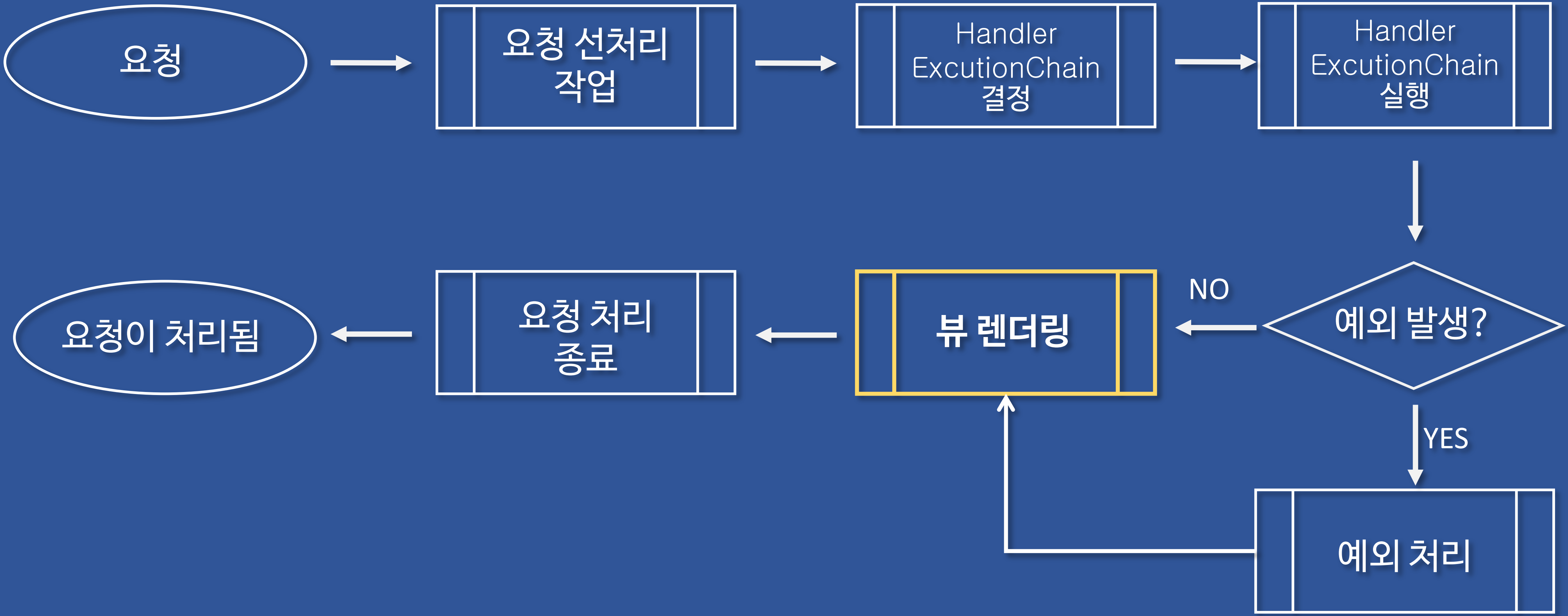
Dispatcher Servlet 내부 요청 처리과정



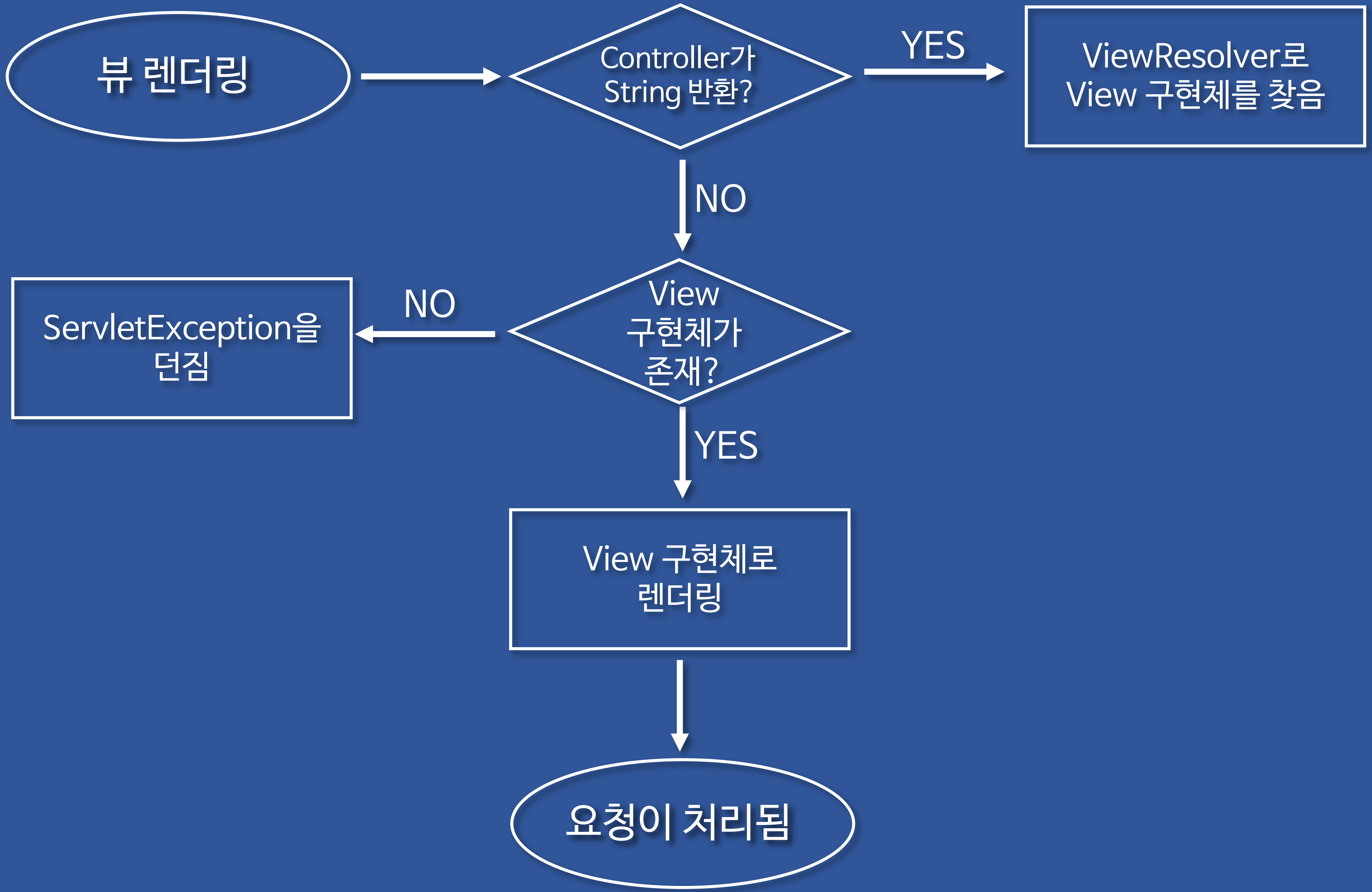
예외 처리


요청 처리 시
예외 발생
HandlerException
Resovler에 문의
ModelAndView
리턴?
YES
요청 처리 재개
NO
예외를 다시 던짐

Dispatcher Servlet 내부 요청 처리과정

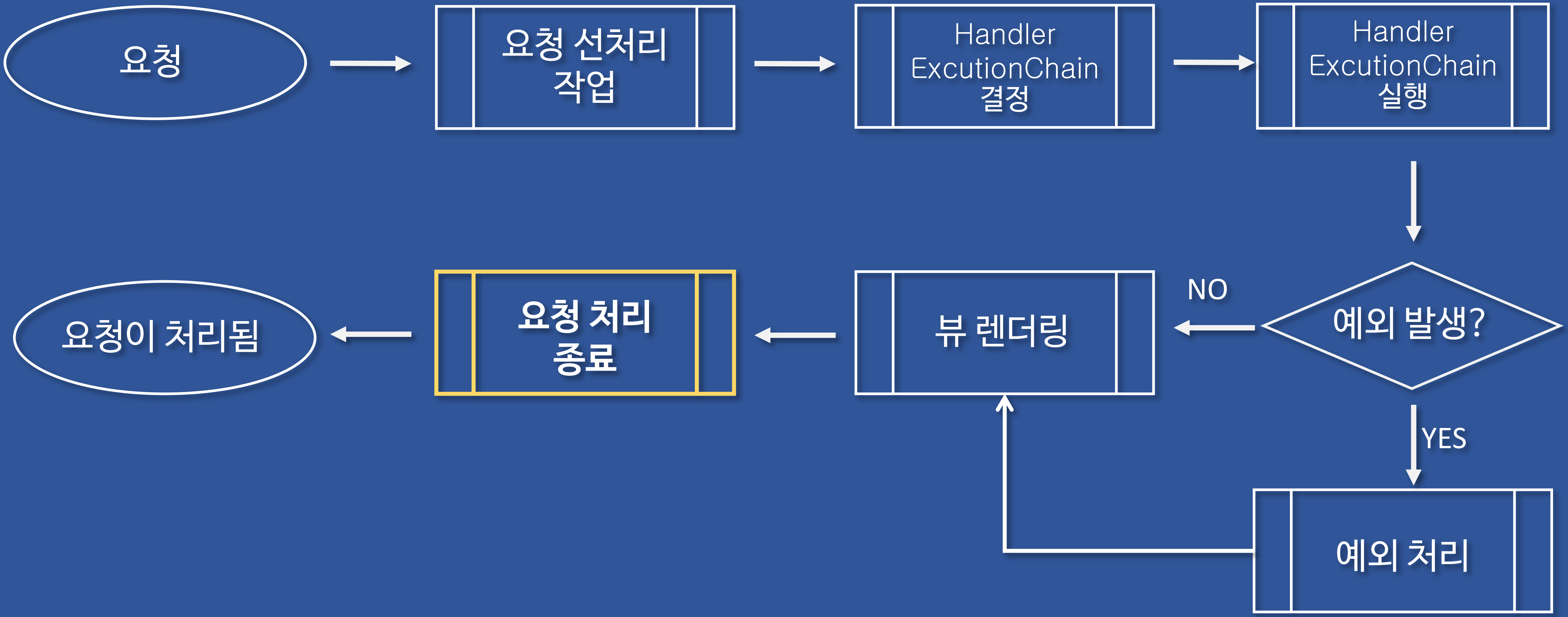


뷰 렌더링 과정

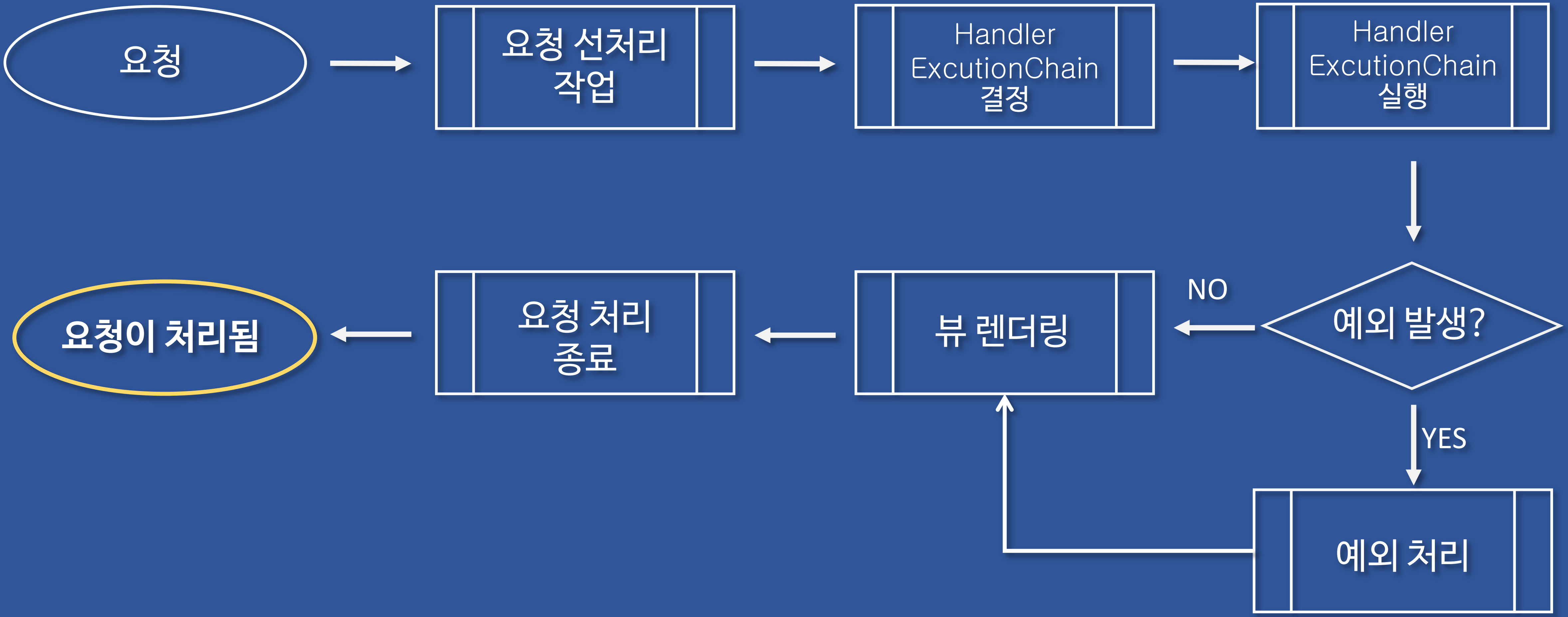


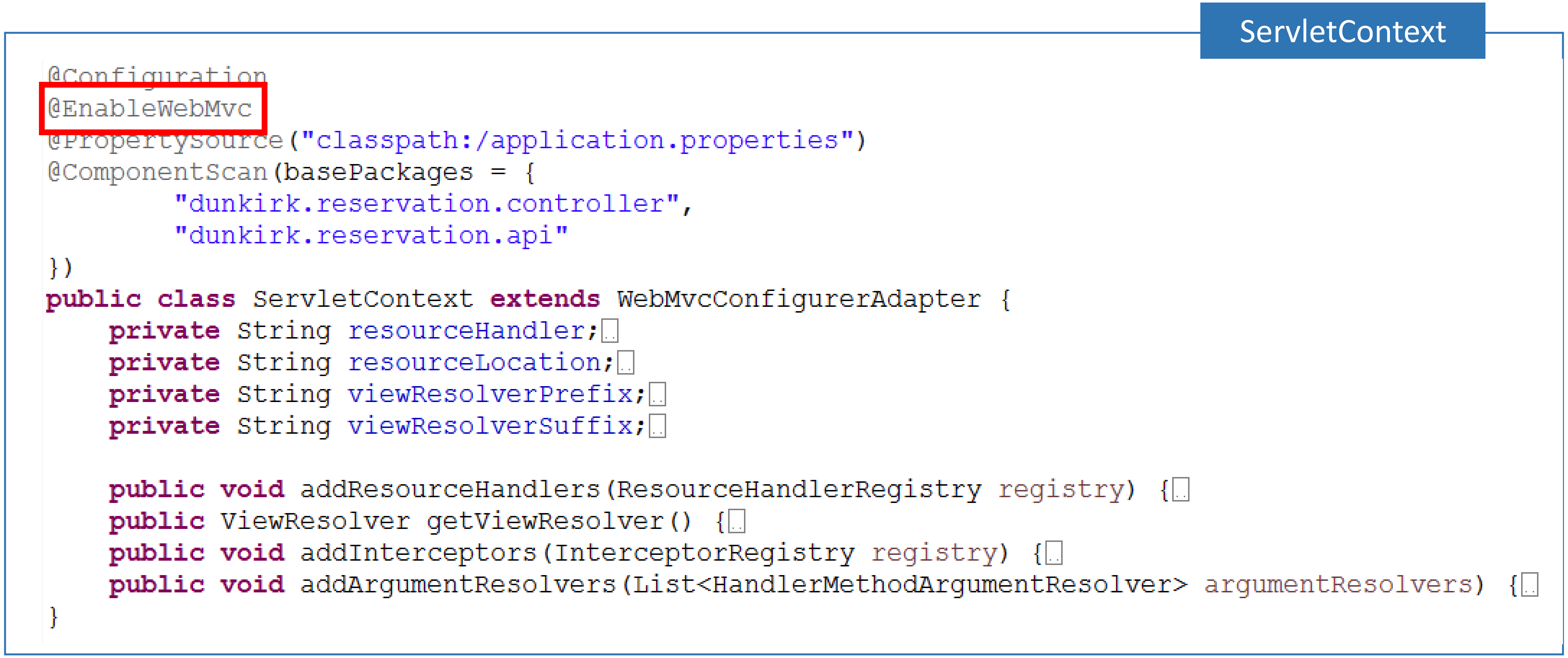
뷰 렌더링
Controller가 String 반환?
YES
ViewResolver로 View 구현체를 찾음
NO
View 구현체가 존재?
NO
ServletException을 던짐
YES
View 구현체로 렌더링
요청이 처리됨

Dispatcher Servlet 내부 요청 처리과정



Dispatcher Servlet 내부 요청 처리과정





ServletContext

```
@Configuration
@EnableWebMvc
@PropertySource("classpath:/application.properties")
@ComponentScan(basePackages = {
        "dunkirk.reservation.controller",
        "dunkirk.reservation.api"
})
public class ServletContext extends WebMvcConfigurerAdapter {
    private String resourceHandler;
    private String resourceLocation;
    private String viewResolverPrefix;
    private String viewResolverSuffix;

    public void addResourceHandlers(ResourceHandlerRegistry registry) {
    public ViewResolver getViewResolver() {
    public void addInterceptors(InterceptorRegistry registry) {
    public void addArgumentResolvers(List<HandlerMethodArgumentResolver> argumentResolvers) {
}
```

@EnableWebMvc

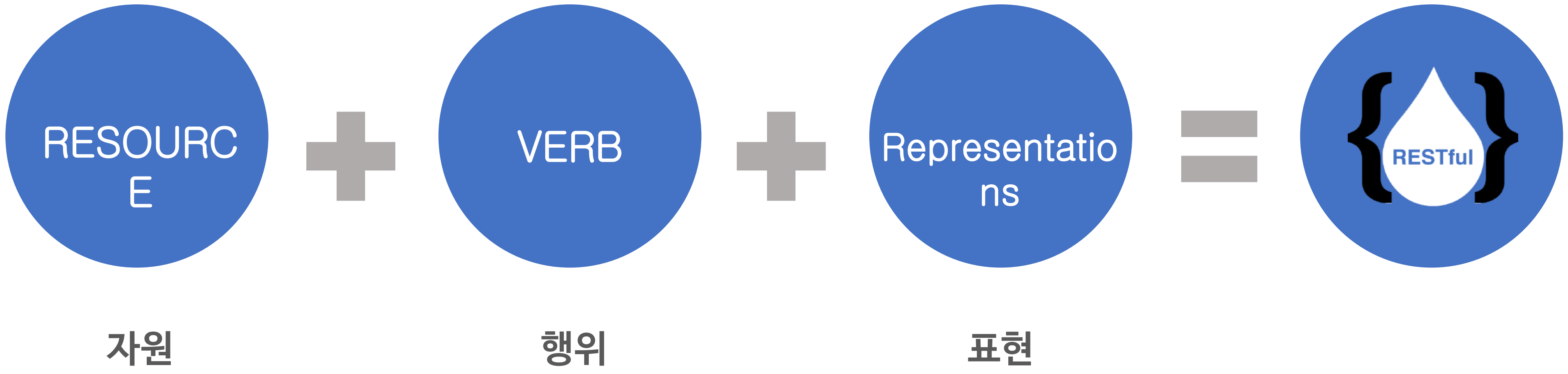
ProductRestController

```
@RestController
@RequestMapping("/products")
public class ProductRestController {
    private ProductService productService;

    @Autowired
    public ProductRestController(ProductService productService) {
        super();
        this.productService = productService;
    }

    @GetMapping
    public List<ProductForMainDto> getList(@RequestParam int categoryId, @RequestParam int page) {
        return productService.getList(categoryId, page * 10);
    }
}
```

{ REST API }

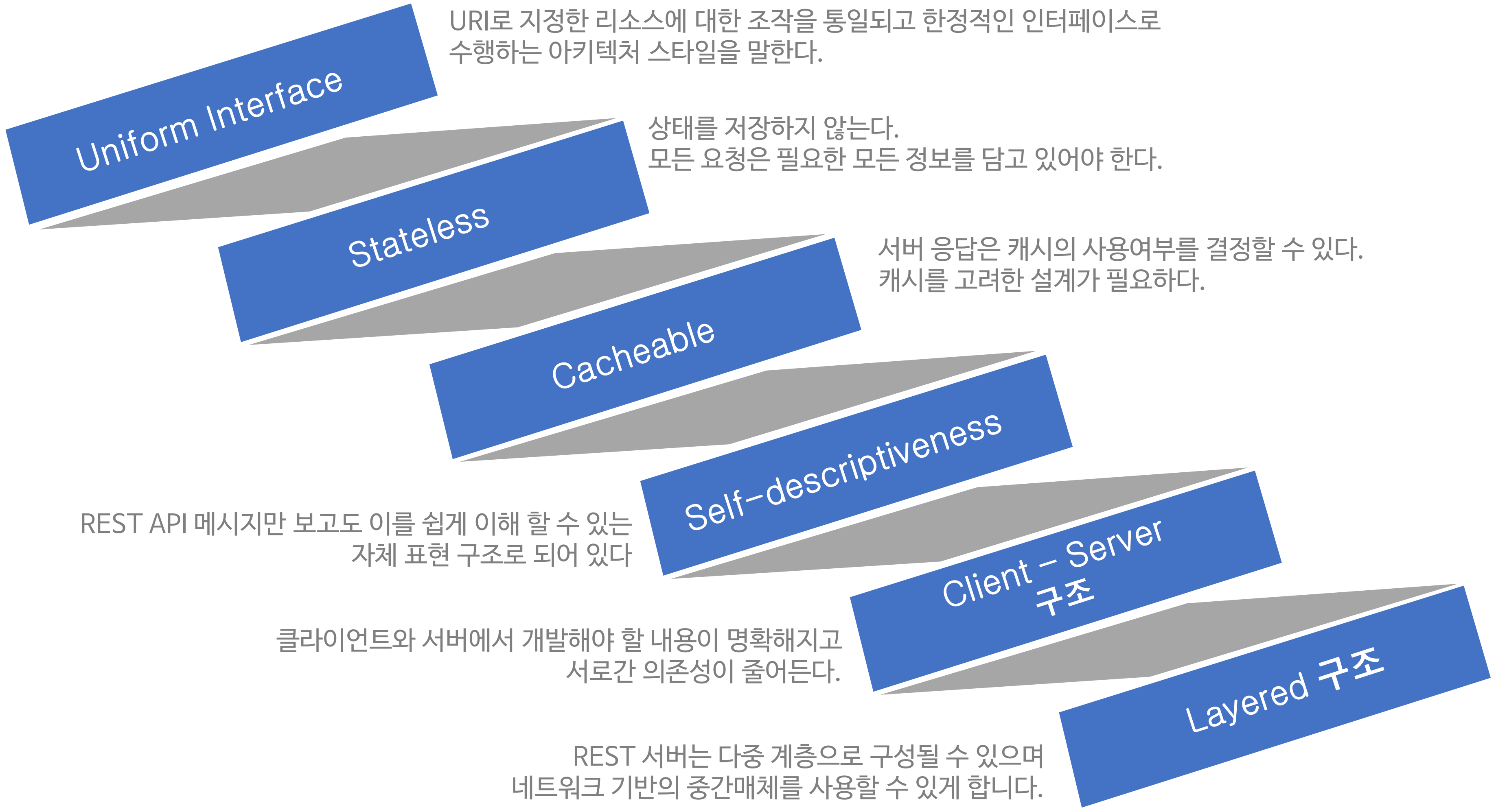
REpresentational State Transfer API

REST API



REST API

REST의 6가지 특징



URI로 지정한 리소스에 대한 조작을 통일되고 한정적인 인터페이스로 수행하는 아키텍처 스타일을 말한다.

상태를 저장하지 않는다.
모든 요청은 필요한 모든 정보를 담고 있어야 한다.

서버 응답은 캐시의 사용여부를 결정할 수 있다.
캐시를 고려한 설계가 필요하다.

REST API 메시지만 보고도 이를 쉽게 이해 할 수 있는 자체 표현 구조로 되어 있다

클라이언트와 서버에서 개발해야 할 내용이 명확해지고 서로간 의존성이 줄어든다.

REST 서버는 다중 계층으로 구성될 수 있으며 네트워크 기반의 중간매체를 사용할 수 있게 합니다.

REST API 디자인 가이드

- URI는 자원을 표시
- 행위는 HTTP Method(GET, POST, PUT, DELETE 등)를 사용
  - GET /sports/soccer/players/delete/2 (X)
  - DELETE /sports/soccer/players/2 (O)

```java
@RestController
@RequestMapping("/comments")
public class CommentRestController {
    private static final int LIMIT_10 = 10;
    private CommentService commentService;

    @Autowired
    public CommentRestController(CommentService commentService) {
        this.commentService = commentService;
    }

    @GetMapping("/{id:[\\d]+}/images")
    public List<Integer> getImageIdList(@PathVariable int id) {
        return commentService.getImageIdList(id);
    }

    @GetMapping
    public List<CommentForDetailDto> getListByProductId(@RequestParam int productId, @RequestParam int page) {
        return commentService.getListByProduct(page, LIMIT_10, productId);
    }

    @PostMapping
    public int add(@RequestBody Comment comment, @AuthUser User user) {
        if(comment != null) {
            comment.setUserId(user.getId());
            comment.setCreateDate(new Timestamp(System.currentTimeMillis()));
            return commentService.add(comment);
        } else {
            return -1;
        }
    }

    @GetMapping("/{id:[\\d]+}")
    public CommentForDetailDto getById(@PathVariable int id) {
        return commentService.getById(id);
    }


    @DeleteMapping("/{id:[\\d]+}")
    public int deleteById(@PathVariable int id) {
        return commentService.deleteById(id);
    }
}
```

[illegible]

1

```java
@GetMapping("/{id:[\\d]+}/images")
public List<Integer> getImageIdList(@PathVariable int id) {
    return commentService.getImageIdList(id);
}
```

2

```java
@GetMapping("/{id:[\\d]+}")
public CommentForDetailDto getById(@PathVariable int id) {
    return commentService.getById(id);
}


@DeleteMapping("/{id:[\\d]+}")
public int deleteById(@PathVariable int id) {
    return commentService.deleteById(id);
}
```

Q&A

감사합니다