Backend Study

Web Application Server

Step 1. Socket, Stream

Step 2. Multi-thread

Step 3. Reflection, Annotation

홍승택

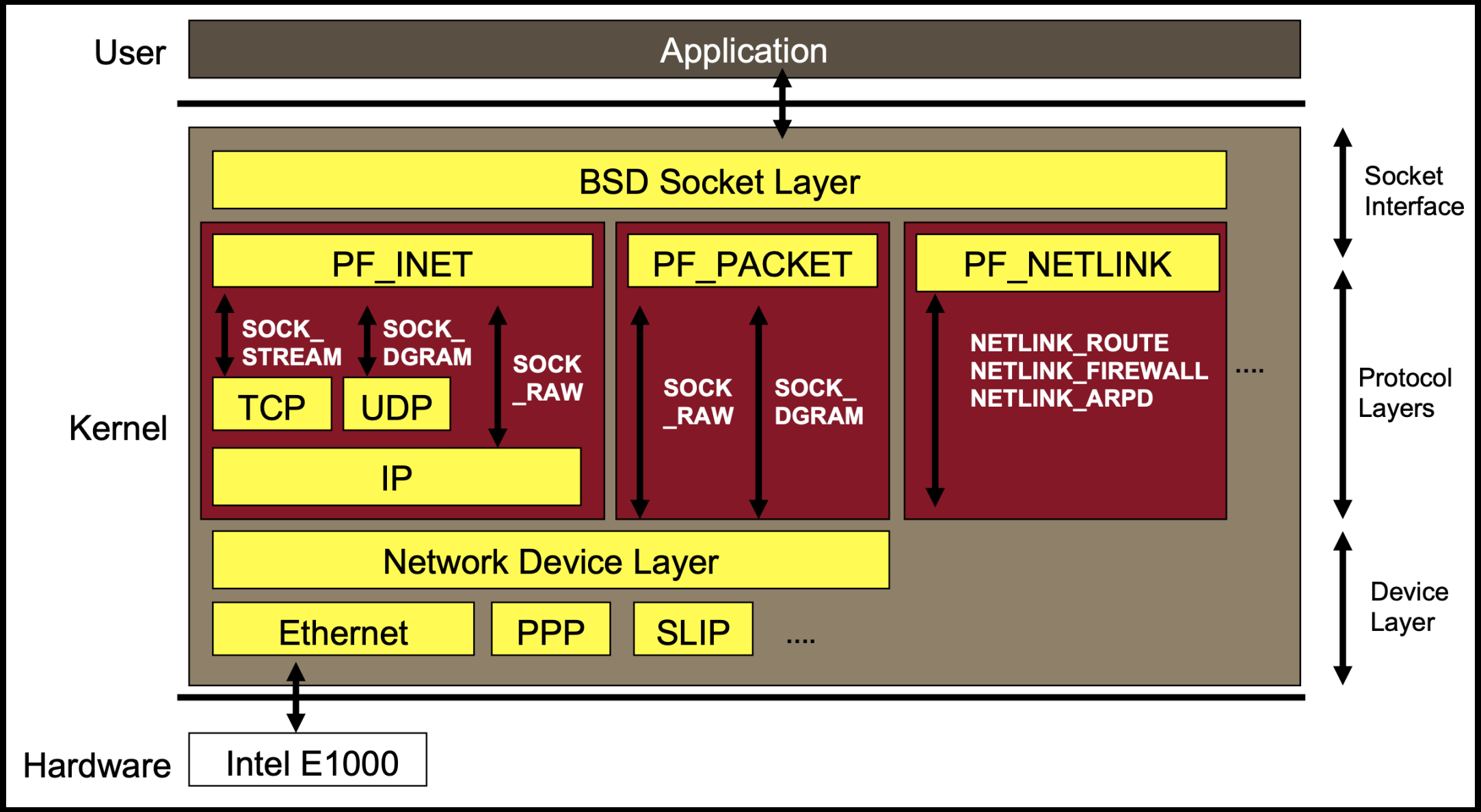
made by Seungtaek

2024. 11. 05. 19시

Step 1.

Socket, Stream

Networking with socket



Socket

Client

```
Socket socket = new Socket("localhost", PORT)
```

localhost의 PORT 포트에 TCP 접속 시도 (blocking)

Server

```
ServerSocket serverSocket = new ServerSocket(port);
```

TCP Listen 시작

```
Socket socket = serverSocket.accept();
```

TCP established 된 소켓 반환 (blocking)

Stream

기본 스트림

```
InputStream input = socket.getInputStream();
OutputStream output = socket.getOutputStream();
```

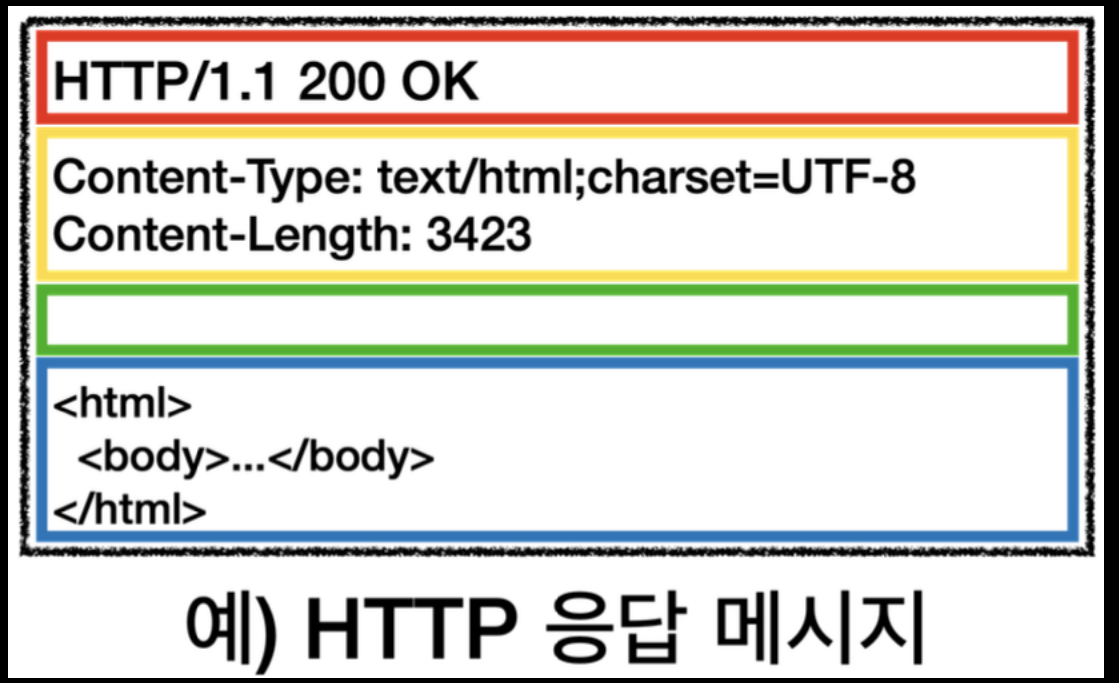
보조스트림: Read

```
InputStreamReader isr = new InputStreamReader(input, UTF_8);
BufferedReader reader = new BufferedReader(isr)
```

보조스트림: Write

```
PrintWriter writer = new PrintWriter(output, false, UTF_8);
```

HTTP Request, Response



HTTP : Basic Server

```
ServerSocket serverSocket = new ServerSocket(port);

while (true) {
    Socket socket = serverSocket.accept();
    process(socket);
}
```

Let’s do coding!

Step 2.

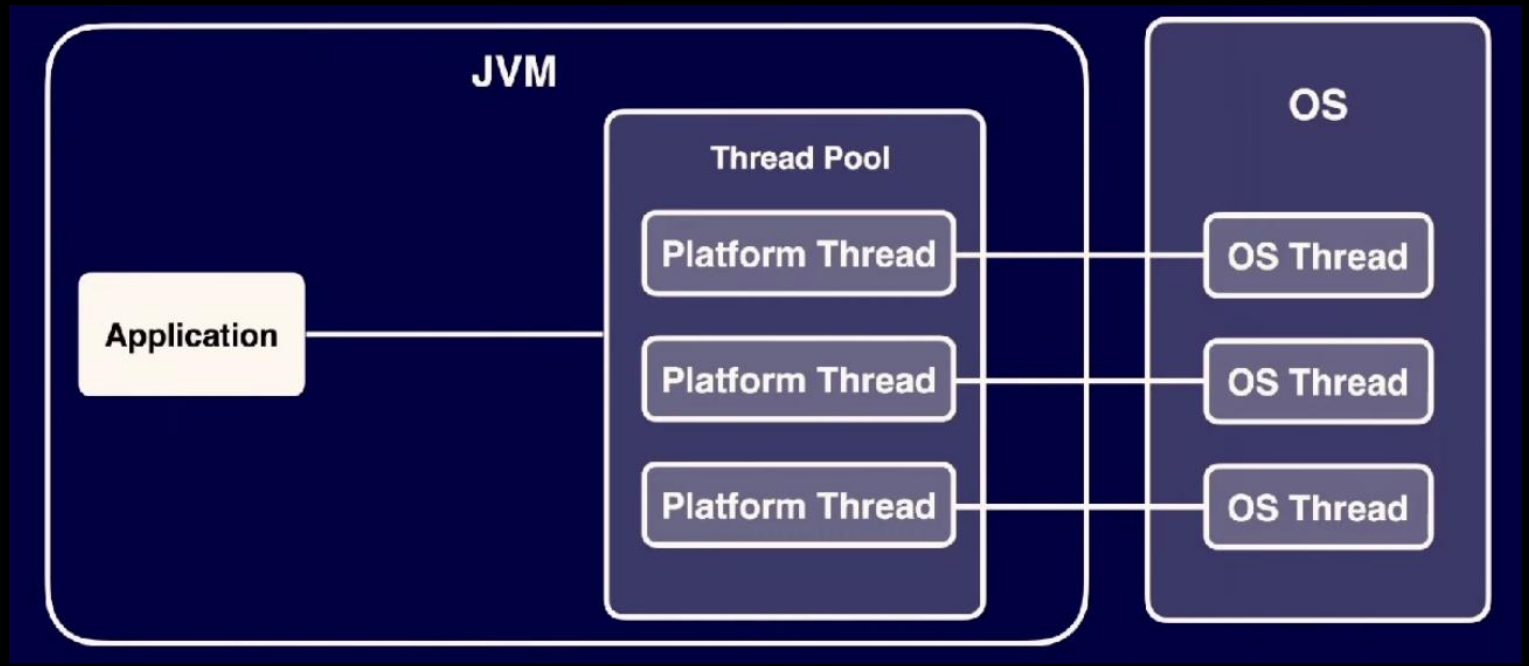
Multi-thread

Java Thread

Java의 Thread는 OS Thread를 Wrapping 한 것 (Platform Thread)

OS Thread는 생성 개수가 제한적이고 생성, 유지하는 비용이 비싸다 -> Thread Pool 사용

```java
// java.lang.Thread.java

private native void start0();

public void start() {
    // ...

    start0();

    // ...
}
```



Java Thread

Runnable 이용

```
Thread thread = new Thread(Runnable target);
thread.start();
```

```
@FunctionalInterface
public interface Runnable {
    When an object implementing interface Runnable is used to create a thread, starting the thread
    causes the object's run method to be called in that separately executing thread.
    The general contract of the method run is that it may take any action whatsoever.
    관련 주제: Thread.run()
    public abstract void run();
}
```

1. Runnable 구현 클래스 선언 및 객체 생성
2. 익명 객체 사용
3. 람다식 이용

Thread 상속

```
public class WorkerThread extends Thread {
  @Override
  public void run() {
     // ...
  }
}

Thread thread1 = new WorkerThread();
thread1.start();


// 익명 자식 객체
Thread thread2 = new Thread() {
    public void run() {
        // ...
    }
}
```

Thread Pool : ExecutorService

메소드명	초기 스레드 수	코어 스레드 수	최대 스레드 수
newCachedThreadPool()	0	0	Integer.MAX_VALUE
newFixedThreadPool(int nThreads)	0	nThreads	nThreads

초기 스레드 수 : ExecutorService 객체가 생성될 때 기본적으로 생성되는 스레드 수

코어 스레드 수 : 사용되지 않는 스레드를 제거할 때 최소한 유지해야 할 스레드 수

최대 스레드 수 : 스레드풀에서 관리하는 최대 스레드 수

```
ExecutorService executorService1 = Executors.newCachedThreadPool();

ExecutorService executorService2 = Executors.newFixedThreadPool(
        Runtime.getRuntime().availableProcessors()
);
```

CPU 코어 수
ex) Apple M1 = 8코어

Thread Pool : ExecutorService

```
ExecutorService threadPool = new ThreadPoolExecutor(
        3,    // 코어 스레드 개수
        100,  // 최대 스레드 개수
        120L, // 놀고 있는 시간
        TimeUnit.SECONDS, // 놀고 있는 시간 단위
        new SynchronousQueue<>() // 작업 큐
);
```

```
public static ExecutorService newCachedThreadPool() {
    return new ThreadPoolExecutor( corePoolSize: 0, Integer.MAX_VALUE,
            keepAliveTime: 60L, TimeUnit.SECONDS,
            new SynchronousQueue<Runnable>());
}
```

```
public static ExecutorService newFixedThreadPool(int nThreads) {
    return new ThreadPoolExecutor(nThreads, nThreads,
            keepAliveTime: 0L, TimeUnit.MILLISECONDS,
            new LinkedBlockingQueue<Runnable>());
}
```

HTTP : Add requirement

1. 스레드풀을 이용한 멀티스레드 사용
2. Request path에 따라 다른 화면 보여주기

```
private final ExecutorService es = Executors.newFixedThreadPool(10);
ServerSocket serverSocket = new ServerSocket(port);

while (true) {
    Socket socket = serverSocket.accept();
    es.submit(new HttpRequestHandler(socket));
}
```

/
/apple
/samsung
/search?model={모델명}

else : 404

HTTP : Percent Encoding

- HTTP URL은 ASCII 만 지원
- UTF-8 문자를 전송하기 위해 퍼센트 인코딩 수행
- 바이트 단위로 묶어서, 16진수 값으로 인코딩 (4+4)

/search?model=가나다 ⟶ /search?model=%EA%B0%80%EB%82%98%EB%8B%A4

UTF-8로 한글을 표현하기 위해서는 3byte 필요
가: EA,B0,80
나: EB,82,98
다: EB,8B,A4

```
String encode = URLEncoder.encode("가", UTF_8);
String decode = URLDecoder.decode(encode, UTF_8);
```

HTTP : HttpRequest, HttpResponse

중복 코드 없애기

부지런히 타이핑

HTTP : Servlet

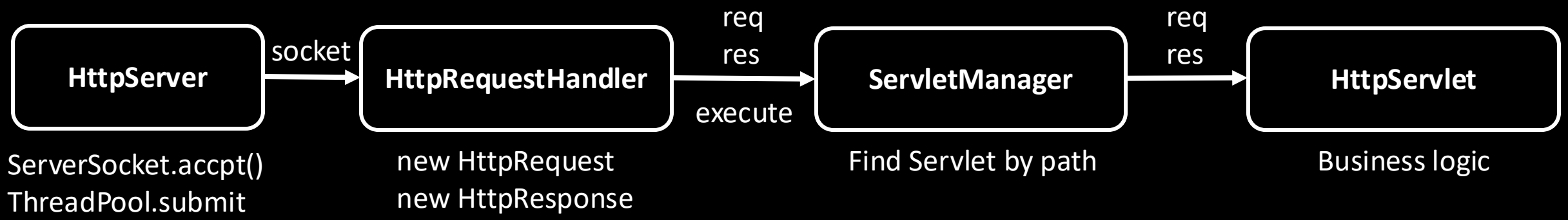
서버간의 호환성이 없음
1990년대 자바 진영에서는 서블릿(Servlet)이라는 표준이 등장

```java
package acompany.server;

import java.io.IOException;

public interface HttpProcess {
    void process(Request request, Response response);
}
```

```java
package bcompany.server;

import java.io.IOException;

public interface HttpCall {
    void call(HttpRequest request, HttpResponse response);
}
```

```java
package jakarta.servlet;
import java.io.IOException;

public interface Servlet {
        void service(ServletRequest var1, ServletResponse var2)
                                throws ServletException, IOException;
        ...
}
```

HTTP : Servlet



HTTP : 요구사항 추가

src / common / repository / ModelRepository, ModelInformation, Store

Apple Store

제조사	제품명	가격
APPLE	iPhone 16 Pro	1550000
APPLE	iMac	1990000
APPLE	MacBook Pro 14	2390000
APPLE	AirPods 4	199000

Home

Samsung Store

제조사	제품명	가격
SAMSUNG	Galaxy Z Flip6	1314000
SAMSUNG	Galaxy S24 Ultra	1919000
SAMSUNG	QLED 4K	2390000
SAMSUNG	Galaxy Book5 Pro 360	2476000

Home

검색 결과

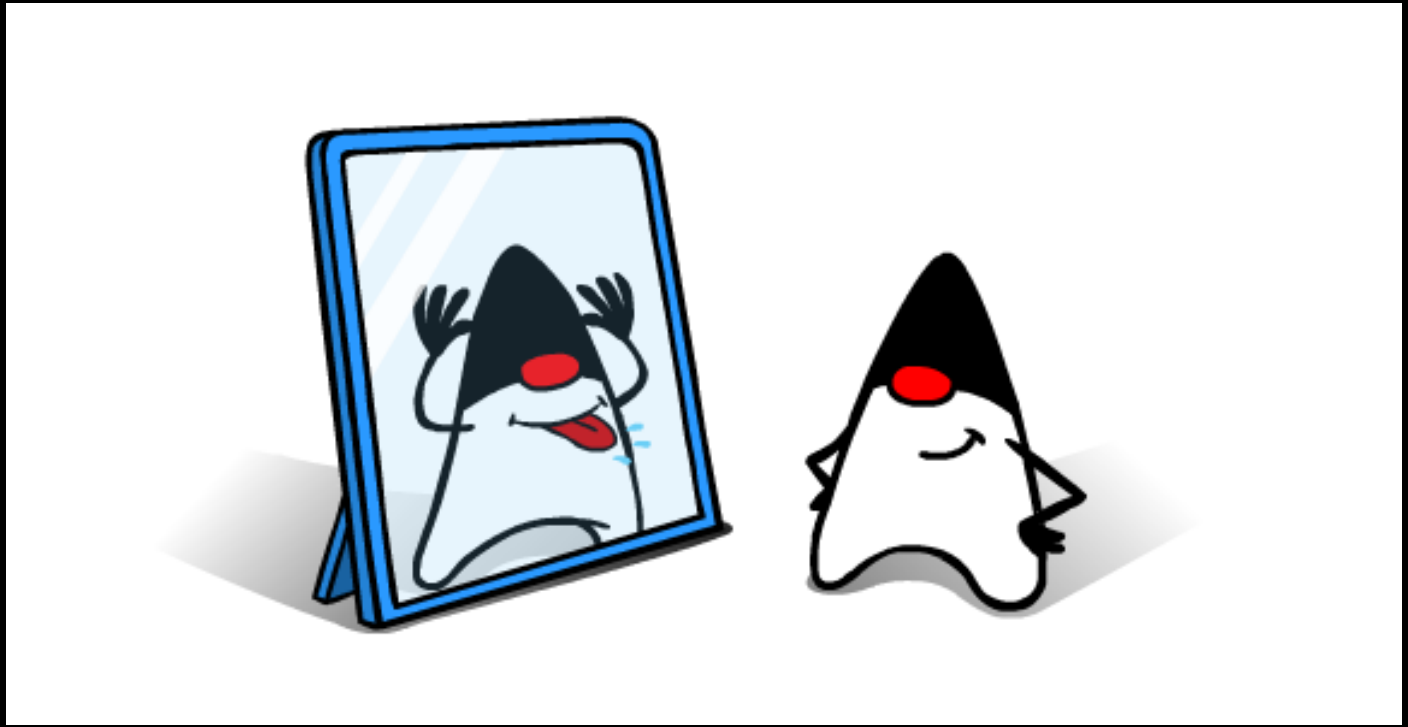
제조사	제품명	가격
APPLE	iPhone 16 Pro	1550000

Home

Step 3.

Reflection, Annotation

Reflection



Annotation

컴파일러나 런타임에서 해석될 수 있는 메타데이터 제공

```
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.METHOD)
public @interface CustomEnno {
    String value();

    int count() default 0;

    String[] tags() default {};
}

@CustomEnno(value = "data", count = 10, tags = {"t1", "t2"})
public void method {
}
```

Reflection + Annotation

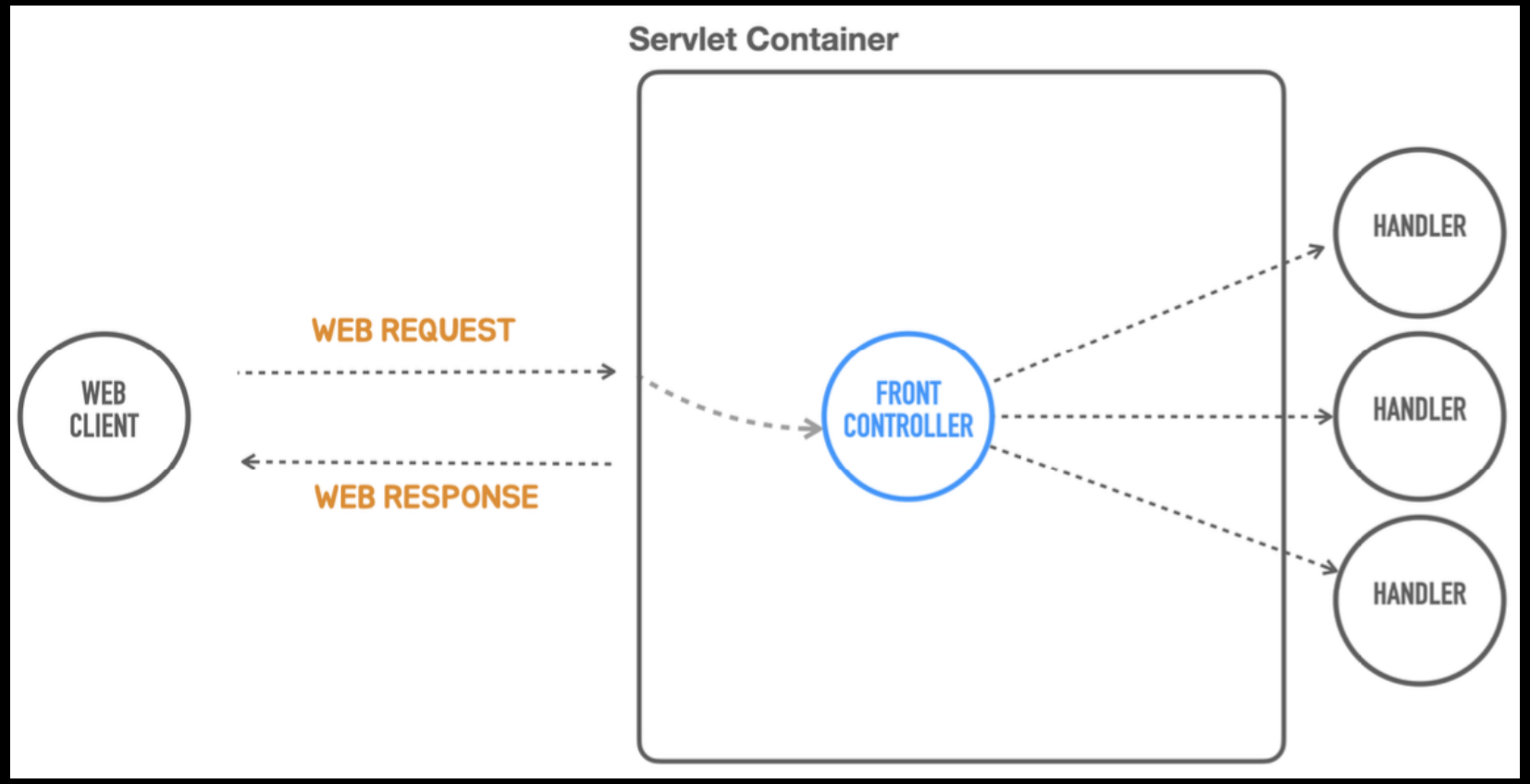
```java
@Retention(RetentionPolicy.RUNTIME)
@Target(ElementType.METHOD)
public @interface GetMapping {
    String value();
}
```

```java
public class CustomController {

    @GetMapping("/request")
    public void search(HttpRequest request, HttpResponse response) {
      // do something
    }
}
```

```java
Method[] methods = controller.getClass().getDeclaredMethods();
for (Method method : methods) {
    if (method.isAnnotationPresent(GetMapping.class)) {
        String path = method.getAnnotation(GetMapping.class).value();
       // path == request.Path ?
    }
}
```

HTTP : Final

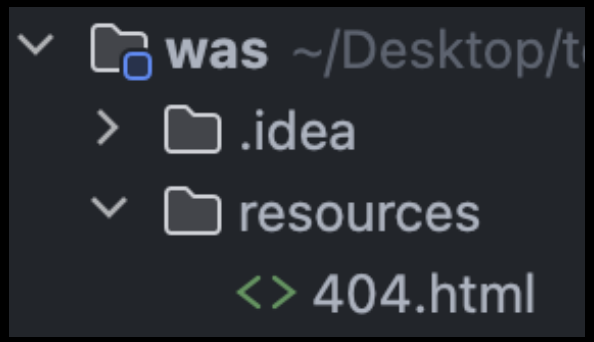
애노테이션을 이용한 Path 매핑



Let’s do coding!

Step 3

HTTP : CustomNotFoundServlet



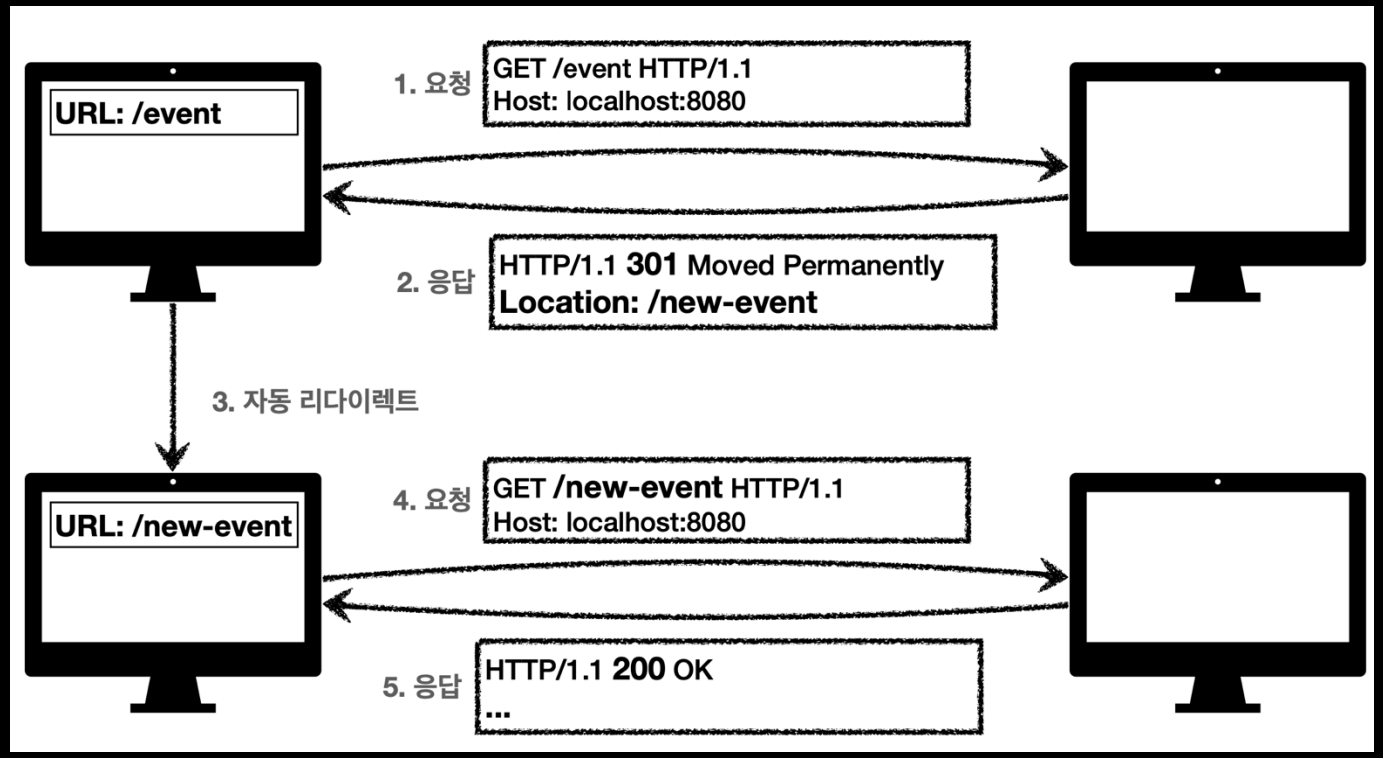

```java
@Override
public void service(HttpRequest request, HttpResponse response) throws IOException {
    response.setStatusCode(404);
    Path path = Path.of( first: "resources/404.html");
    try (Stream<String> lineStream = Files.lines(path, StandardCharsets.UTF_8)) {
        lineStream.forEach(response::writeBody);
    }
}
```

HTTP : Challenge

Redirect 기능 추가하기 (RedirectController.java)
/redirect 로 요청 시, /apple 로 redirect

301 Moved Permanently
- 영구 리다이렉션
- 리다이렉트시 요청 메서드가 GET으로 변하고, 본문이 제거될 수 있음(MAY)

302 Found
- 일시적 리다이렉션
- 리다이렉트시 요청 메서드가 GET으로 변하고, 본문이 제거될 수 있음(MAY)



Reference

김영한의 실전 자바 - 고급 2편, I/O, 네트워크, 리플렉션