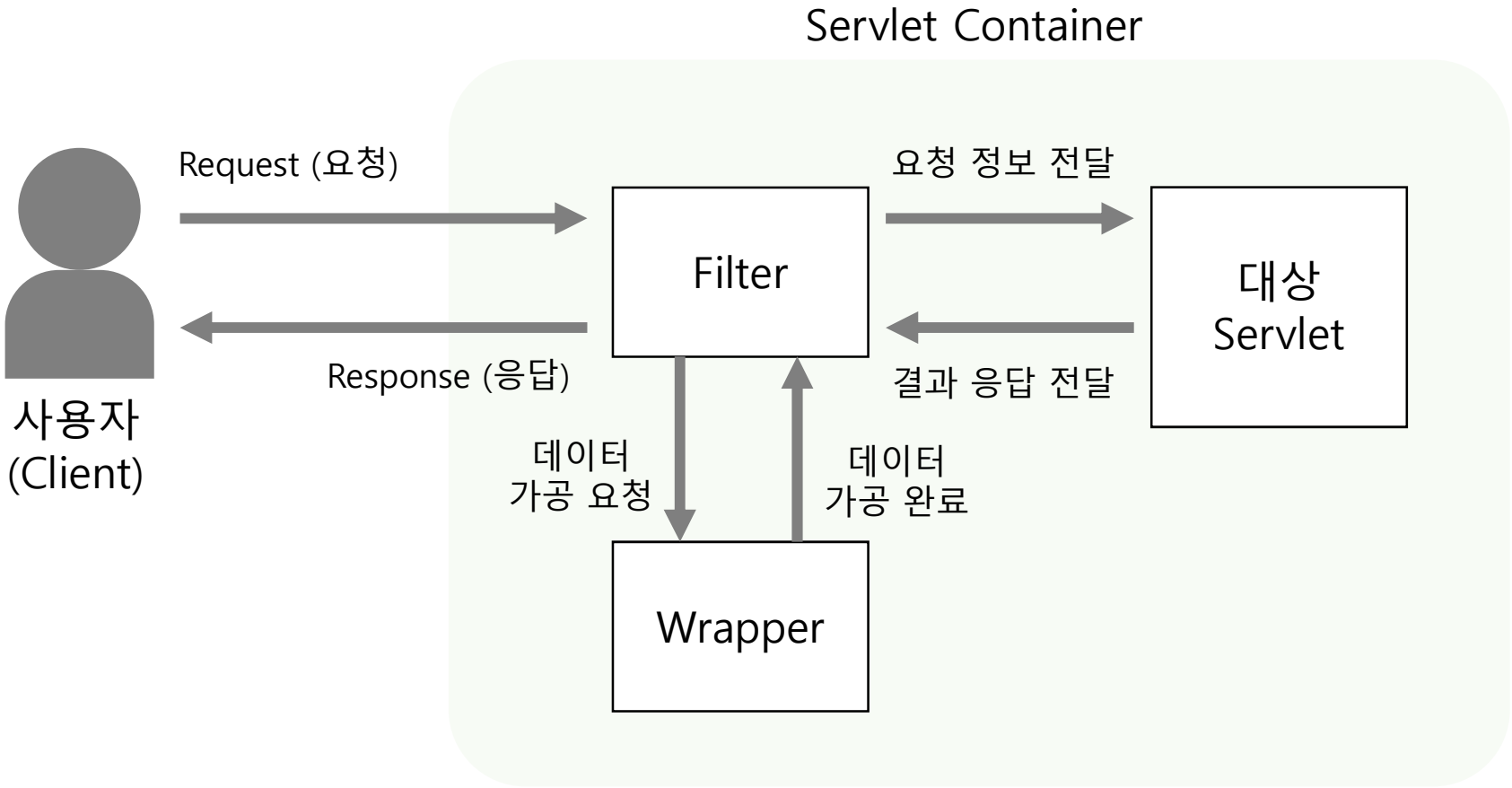
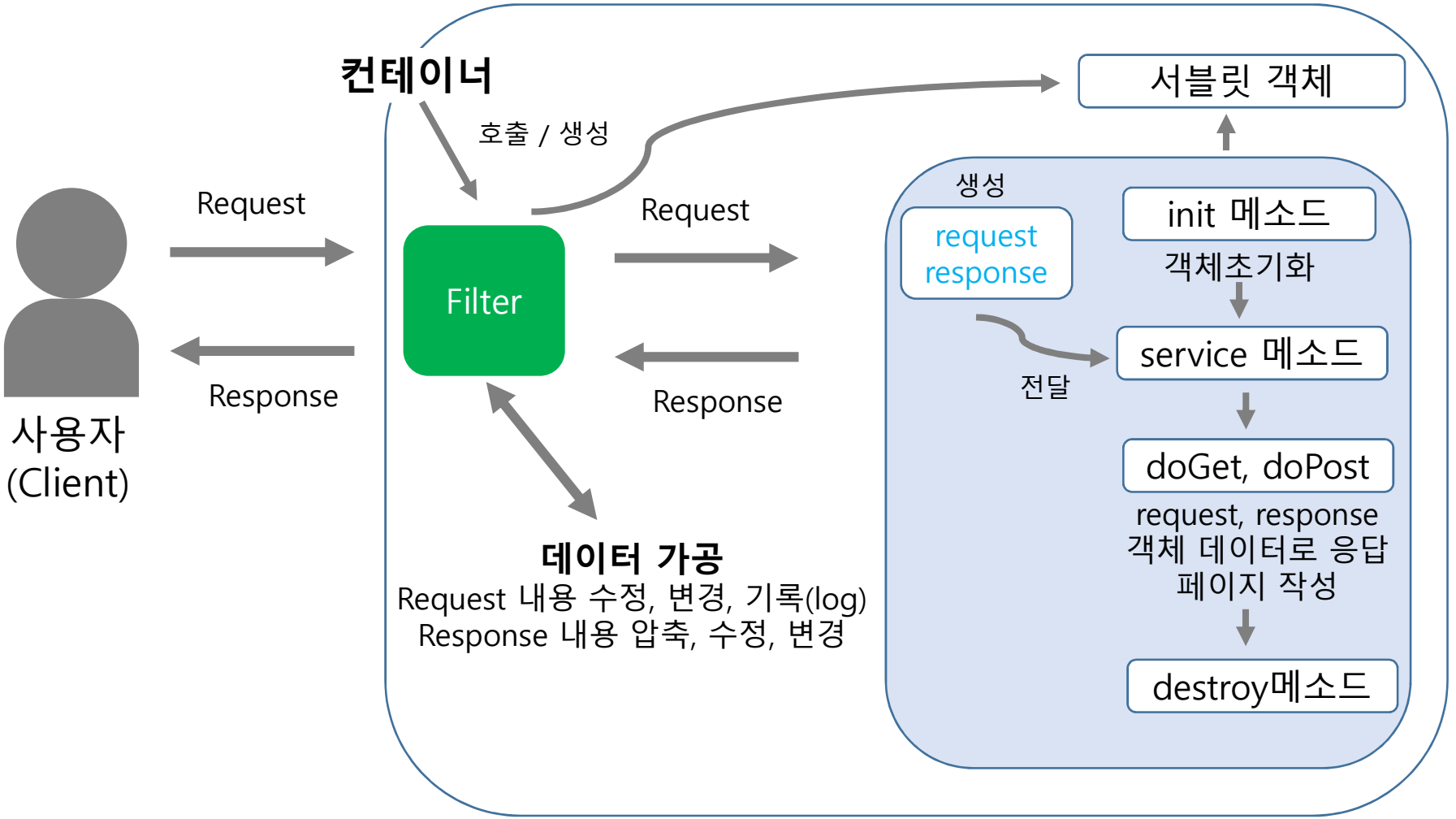
Filter와 Wrapper

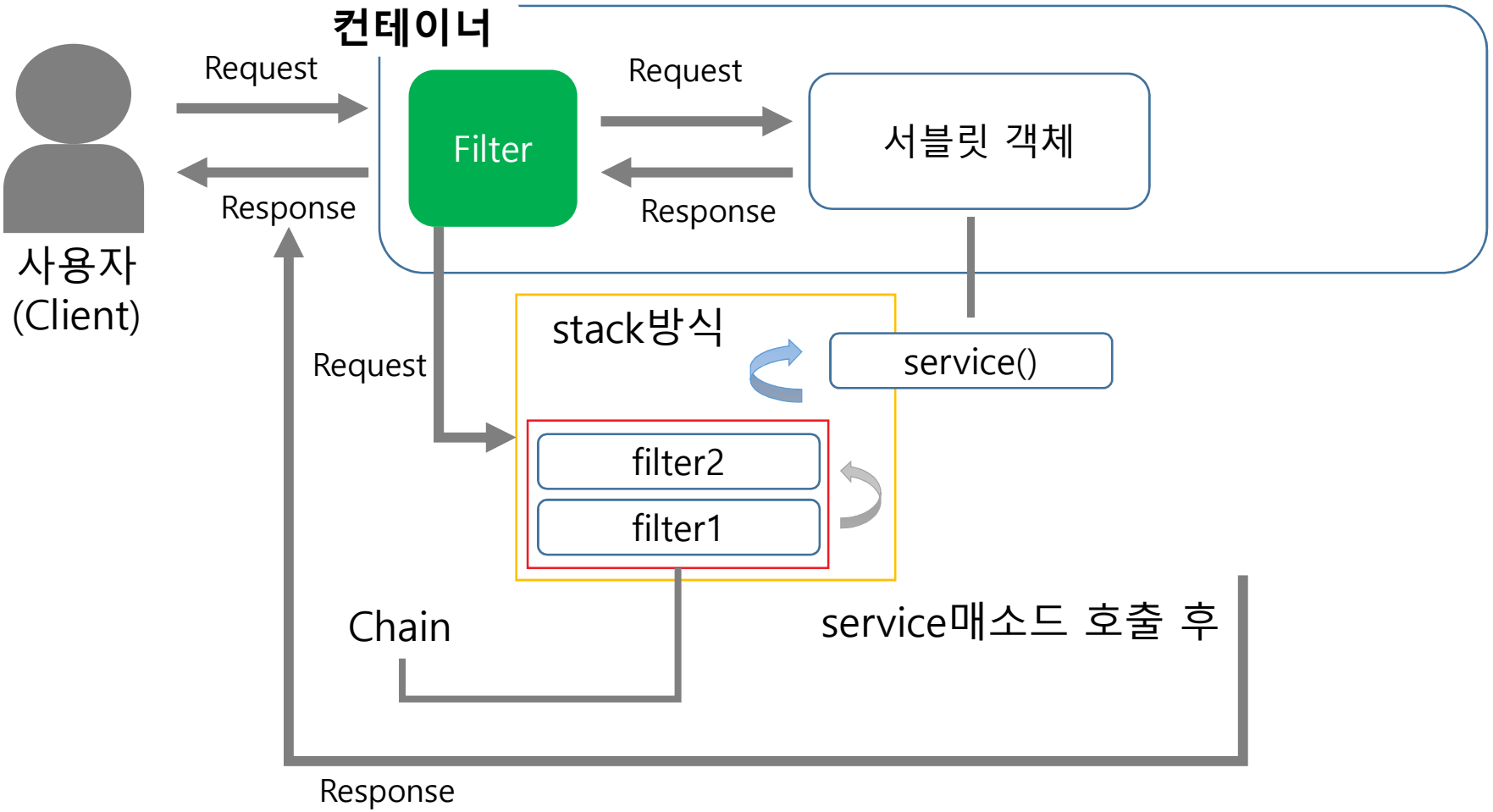
▶ 서블릿 필터와 래퍼 동작 구조



▶ 서블릿 필터 동작 구조


컨테이너
호출 / 생성
서블릿 객체
Request
Request
Filter
Response
Response
사용자
(Client)
생성
request
response
init 메소드
객체초기화
service 메소드
전달
doGet, doPost
request, response
객체 데이터로 응답
페이지 작성
destroy메소드
데이터 가공
Request 내용 수정, 변경, 기록(log)
Response 내용 압축, 수정, 변경

▶ 서블릿 필터 내부 동작


컨테이너
Request
Filter
Request
서블릿 객체
Response
Response
사용자
(Client)
stack방식
service()
Request
filter2
filter1
Chain
service매소드 호출 후
Response

▶ Servlet Filter & Wrapper

✓ 서블릿 필터

javax.servlet.Filter인터페이스를 상속받아 구현하는 클래스
HTTP요청과 응답 사이에서 전달되는 데이터를 가로채 서비스에 맞게
수정하는 필터링 작업을 수행할 수 있는 클래스
웹 브라우저가 필요한 서블릿을 호출할 경우,
필터가 대신 호출되어 전달받은 정보를 수정하고
서블릿에게 넘기는 일종의 '경유지'역할
Servlet과 init이나 destroy, doGet처럼 비슷한 모습을 보이며
Request는 보안 관련 사항, 요청 헤더와 바디 형식 지정,
요청에 대한 log 기록 유지 등을 처리하고
Response는 응답 스트림 압축, 응답 스트림 내용 추가 및 수정,
새로운 응답 작성 등 처리

▶ filter설정하기

DD설정(web.xml)

Filter등록

```
<filter>
    <filter-name>필터설정이름</filter-name>
    <filter-class>필터를 구현한 클래스</filter-class>
    <init-param> // filter에서 사용할 값 설정
      <param-name>초기값설정이름</param-name>
       <param-value>설정값</param-value>
    </init-param>
 </filter>
```

▶ filter설정하기

DD설정(web.xml)

Filter-Mapping(url패턴과 매핑)

```
<filter-mapping>
    <filter-name>등록된 필터이름</filter-name>
    <url-pattern>요청한 페이지 형식</url-pattern>
</filter-mapping>
```

서블릿과 필터 맵핑(필터 적용 서블릿지정)

```
<filter-mapping>
    <filter-name>등록된 필터이름</filter-name>
    <servlet-name>적용할 서블릿명</servlet-name>
</filter-mapping>
```

** 매핑하는방법이 두가지 / url-pattern이 우선적용

filter설정하기 - java

```
public class 클래스명 implements Filter
{
    @Override
     public void init(FilterConfig config) throws ServletException
     {          Filter호출시 작업 설정       }
     @Override
     public void doFilter(ServletRequest req, ServletResponse resp,
                          FilterChain chain) throws ServletException,
                                                    IOException
      {          필터링 작업할 내용          }
     @Override
     public void destroy()
      { 삭제시 작업 설정                    }
}
```

▶ Servlet Filter & Wrapper

✓ Filter Interface

- init(FilterConfig config)

  웹 컨테이너가 필터를 호출할 경우 해당 메소드가 호출되어 필터 객체를 생성하며 초기화

  * 매개변수 FilterConfig는 web.xml에 있는 <filter>정보를 가지고 있음

- doFilter(ServletRequest req, ServletResponse res, FilterChain chain)

  필터가 수행될 때 구동하는 메소드로,
  요청 객체와 응답 객체를 사용해 일련의 작업을 수행한 뒤 chain을 통해 가공된 값을 목적지로 전송

- destroy()

  역할이 끝난 필터는 웹 컨테이너에 의해 해당 메소드를 호출하고 소멸

▶ Servlet Filter 구현 예

✓ java파일 구현

```
package test.common.filter;

import java.io.IOException;

public class CharsetEncodingFilter implements Filter {

    @Override
    public void init(FilterConfig config) throws ServletException {}

    @Override
    public void doFilter(ServletRequest req, ServletResponse res, FilterChain chain)
            throws IOException, ServletException {

        req.setCharacterEncoding("UTF-8");
        res.setContentType("text/html; charset=UTF-8");

        chain.doFilter(req, res);

    }

    @Override
    public void destroy() {}
}
```

▶ Servlet Filter 구현 예

✓ web.xml파일 설정

```
<filter>
 <filter-name>EncodingFilter</filter-name>
 <filter-class>test.common.filter.CharsetEncodingFilter</filter-class>
 <init-param>
     <param-name>encoding</param-name>
     <param-value>UTF-8</param-value>
 </init-param>
</filter>

<filter-mapping>
 <filter-name>EncodingFilter</filter-name>
 <url-pattern>/*</url-pattern>
</filter-mapping>
```

▶ 서블릿 필터 참고

✓ interface FilterChain

필터는 chain처럼 서로 연결되어 있는데 연결 되어있는 필터를
순차 별로 doFilter()메소드를 이용해 실행시키는 인터페이스
마지막 필터가 필터가 실행된 후에는 service()메소드를 실행시켜
서블릿의 메소드(doGet(), doPost()) 실행

* doFilter(ServletRequest req, ServletResponse)
chain으로 연결되어 있는 다음 필터를 실행하는 메소드

▶ Servlet Filter & Wrapper

✓ 서블릿 래퍼

필터 클래스로부터 전달받은 데이터를 가공하여 다시 필터에게 반환하는 클래스

데이터 가공이 필요한 시점이 요청일 경우

HttpServletRequestWrapper클래스를 통해 구현하고

응답일 경우 HttpServletResponseWrapper클래스를 통해 구현

▶ Servlet Filter & Wrapper

✓ Wrapper Class

- HttpServletRequestWrapper

요청한 정보를 변경하는 Wrapper클래스
HttpServletRequest객체를 매개로 하는 생성자를 가짐

```
public SampleWrapper(HttpServletRequest wrapper){
        super(wrapper);
}
```

- HttpServletResonseWrapper

응답할 정보를 변경하는 Wrapper클래스
HttpServletResponse객체를 매개로 하는 생성자를 가짐

```
public SampleWraper(HttpServletResponse wrapper){
        super(wrapper);
}
```

▶ 서블릿 필터와 래퍼 클래스 사용 사례

서블릿 필터와 래퍼 클래스를 사용해 다음과 같은 로직 구현 가능

1. 서비스 별 로그 기록
2. 이미지 변환 필터
3. 문자셋 변환 필터
4. 비밀번호 암호화 로직 구현
5. 사용자 인증 로직 구현