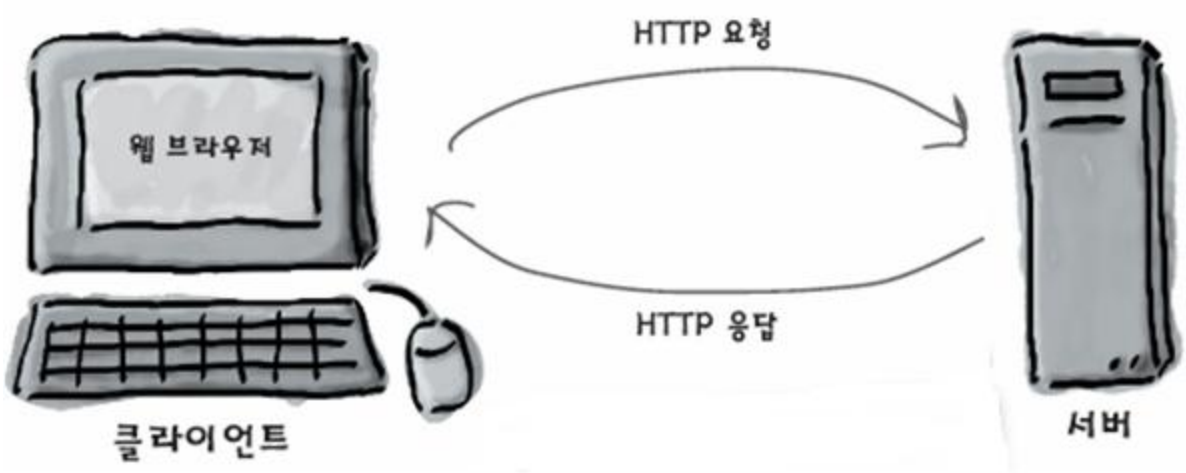
3. Servlet

HTTP 프로토콜

- HTTP 프로토콜 : TCP/IP 를 기반으로 하여 웹에서 사용하는 프로토콜로서 요청(Request) 과 응답(Response) 데이터를 전송하는 방식



HTTP 요청과 HTTP 응답

- HTTP 요청(Request) 주요 구성요소
  - HTTP 메소드 (실행할 액션)
  - 접근하고자 하는 URL
  - 폼 파라미터 (메소드의 매개변수와 비슷함)

- HTTP 응답(Response) 주요 구성요소
  - 상태코드 (요청 처리에 대한 성공여부)
  - 컨텐츠 타입 (텍스트, 그림, HTML 등)
  - 컨텐츠 (HTML 코드, 이미지 등)

HTTP 요청방식

- GET 방식
  - 서버에 있는 정보를 가져오기 위해 설계됨.
  - 240바이트까지 전달할 수 있음.
  - QUERY_STRING 환경변수를 통해 전달.
  - 형식 : http://xxx.xxx.co.kr/servlet/login?id=hj&name=hong
  - URL노출로 보안이 요구되는 경우엔 사용할 수 없음.
  - 검색엔진에서 검색단어 전송에 많이 이용함.

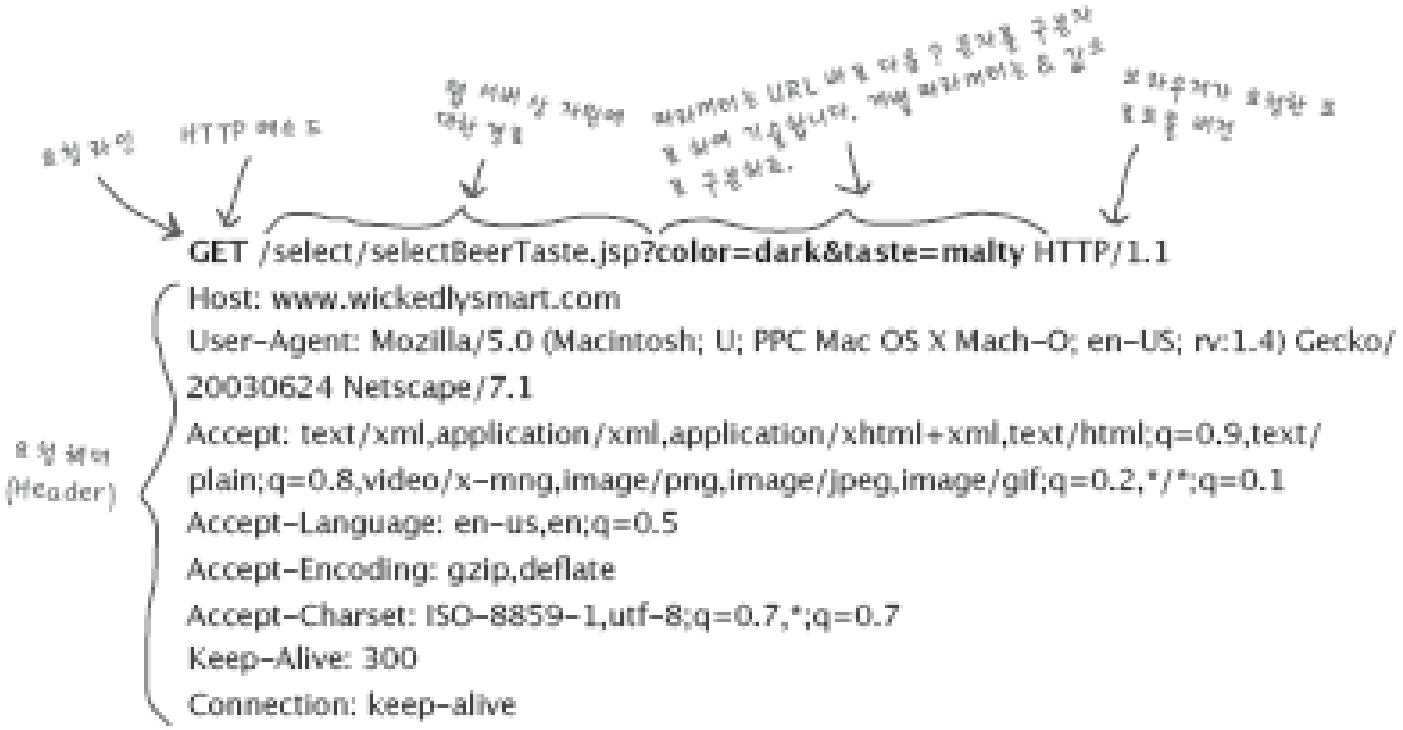
- POST방식
  - 서버로 정보를 올리기 위해 설계됨.
  - 데이터크기의 제한은 없다.
  - URL에 파러미터가 표시되지 않는다.

HTTP 요청방식– GET 방식

- HTTP 메소드 : HTTP 요청의 종류 및 폼 파라미터의 포맷을 정의. 주로 사용하는 메소드는 GET, POST 가 있음
- GET 메소드 : 단순한 자원(HTML 문서, 이미지 등) 요청. 간단한 파라미터 정도만 HTTP 요청 시 사용

HTTP 요청방식– GET 방식

- GET 방식일 때의 요청 메시지의 내용

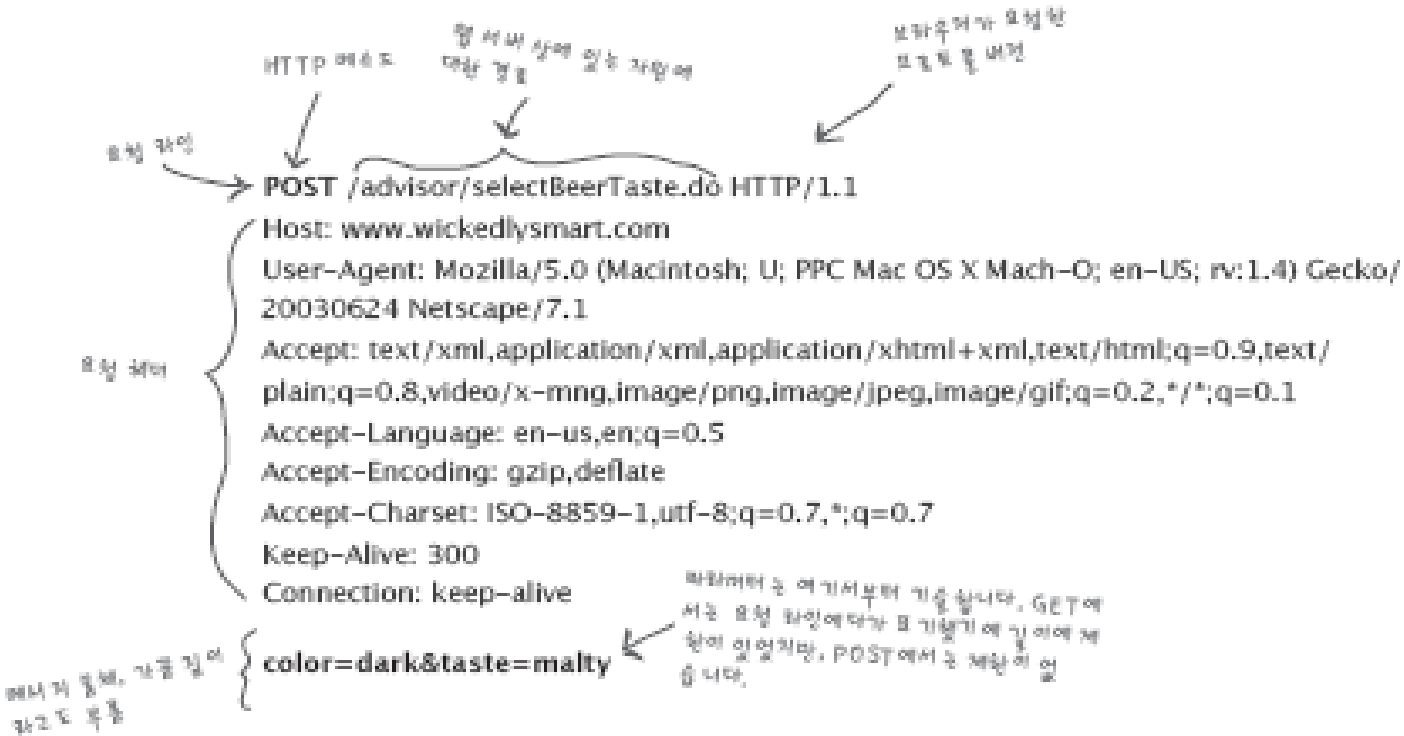


HTTP 요청방식– POST 방식

- POST 메소드 : 사용자의 입력값을 HTTP 요청 시 서버에 전달. 복잡한 파라미터 사용 가능

HTTP 요청방식– POST 방식

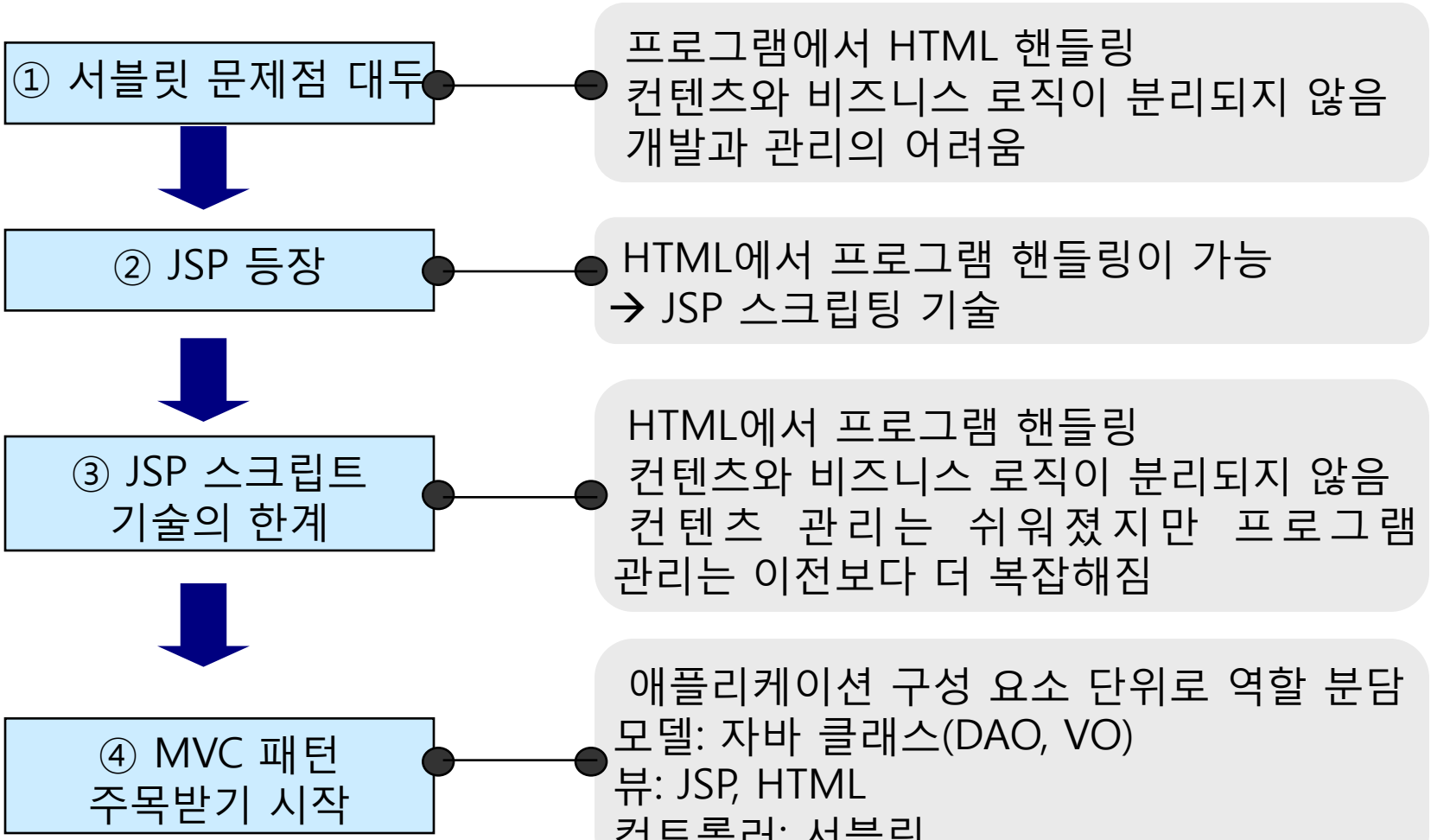
- POST 방식일 때의 요청 메시지의 내용



Servlet 개요

- 자바 플랫폼에서 컴포넌트 기반의 웹 애플리케이션 개발기술
- JSP는 서블릿 기술에 기반함
- 서블릿의 프리젠테이션 문제를 해결하기 위해 JSP가 등장
- JSP 모델2 가 주목받으며 다시 서블릿에 대한 중요성 부각.

Servlet 개요

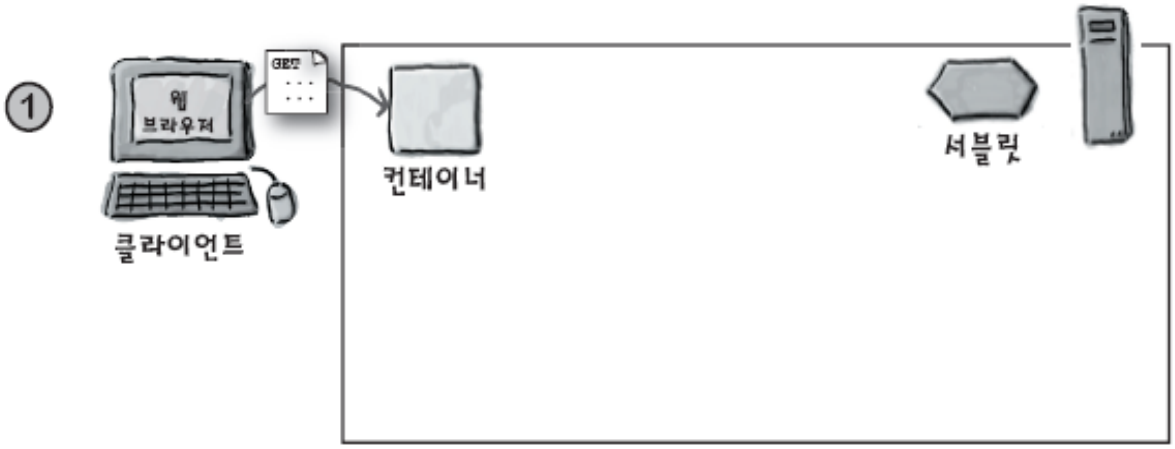

① 서블릿 문제점 대두
프로그램에서 HTML 핸들링
컨텐츠와 비즈니스 로직이 분리되지 않음
개발과 관리의 어려움
② JSP 등장
HTML에서 프로그램 핸들링이 가능
→ JSP 스크립팅 기술
③ JSP 스크립트 기술의 한계
HTML에서 프로그램 핸들링
컨텐츠와 비즈니스 로직이 분리되지 않음
컨텐츠 관리는 쉬워졌지만 프로그램 관리는 이전보다 더 복잡해짐
④ MVC 패턴 주목받기 시작
애플리케이션 구성 요소 단위로 역할 분담
모델: 자바 클래스(DAO, VO)
뷰: JSP, HTML
컨트롤러: 서블릿

Web Container

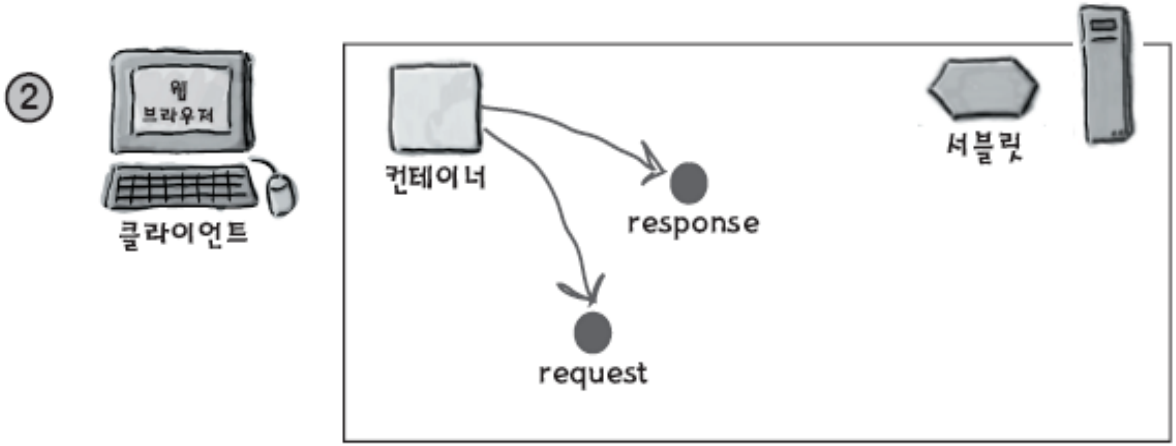
- 자바가상머신을 내장한 서블릿 운영환경
- JSP는 서블릿으로 변환되어 실행

  → 따라서 대부분 별도의 실행환경 없이 웹 컨테이너에 통합됨.
- 자체 웹 서버 기능도 있으나 웹 서버와 분리하기도 함.
- 대표적으로 apache tomcat이 있음.

Servlet 라이프 사이클



사용자가 서블릿에 대한 링크(URL)를 클릭합니다.



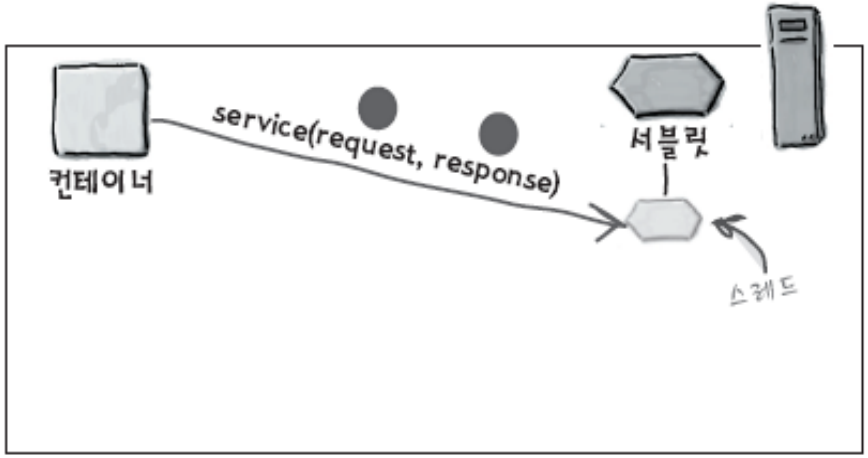
컨테이너는 요청된 Request가 서블릿이라는 것을 간파하고는 다음 두 개의 객체를 생성합니다.

1) HttpServletResponse

2) HttpServletRequest

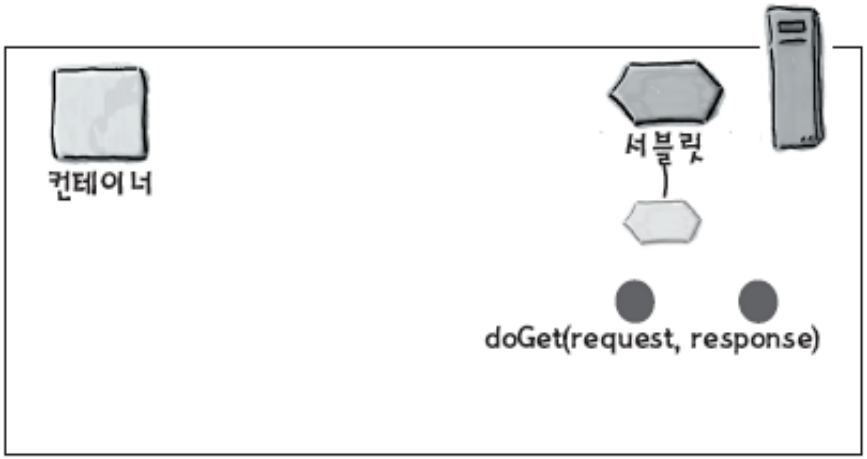
Servlet 라이프 사이클

③



접수한 요청의 URL을 분석하여 어떤 서블릿을 요청했는지 파악합니다(여기서 DD를 참조하지요). 그 다음 해당 서블릿 스레드를 생성하여 Request, Response 객체 참조를 넘깁니다.

④

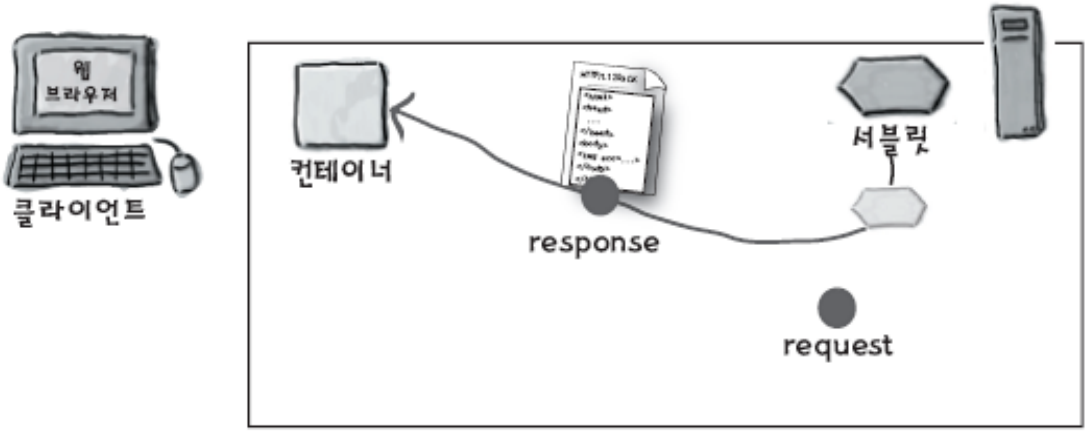


컨테이너는 서블릿 service() 메소드를 호출합니다. 브라우저에서 지정한 방식에 따라 doGet()을 호출할지, doPost()를 호출할지 결정합니다.

클라이언트가 HTTP GET 메소드를 날렸다면, service() 메소드는 서블릿의 doGet() 메소드를 호출합니다. 호출할 때 Request와 Response 객체를 인자로 넘깁니다.

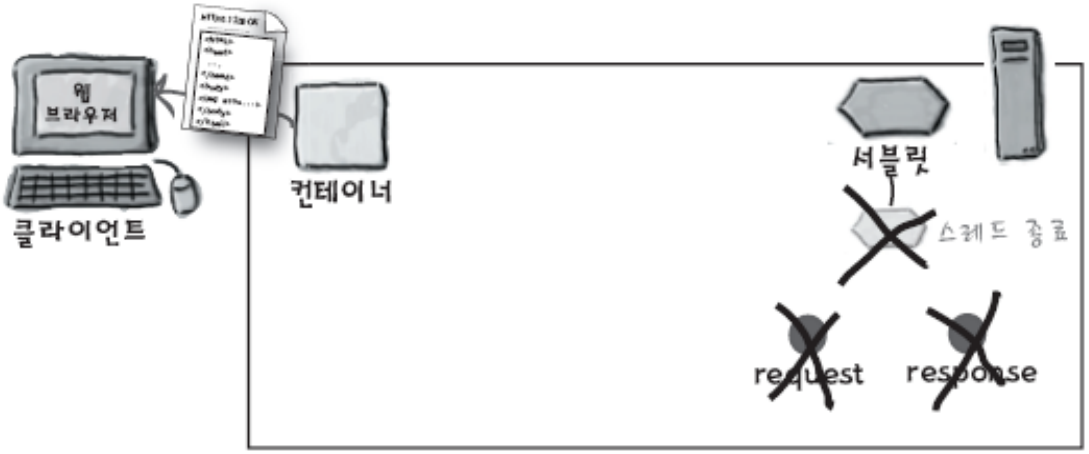
Servlet 라이프 사이클

⑤



서블릿은 클라이언트에게 응답을 작성(write)하기 위하여 Response 객체를 사용합니다. 이 작업을 완료하면, Response에 대한 제어는 컨테이너에게 넘어갑니다.

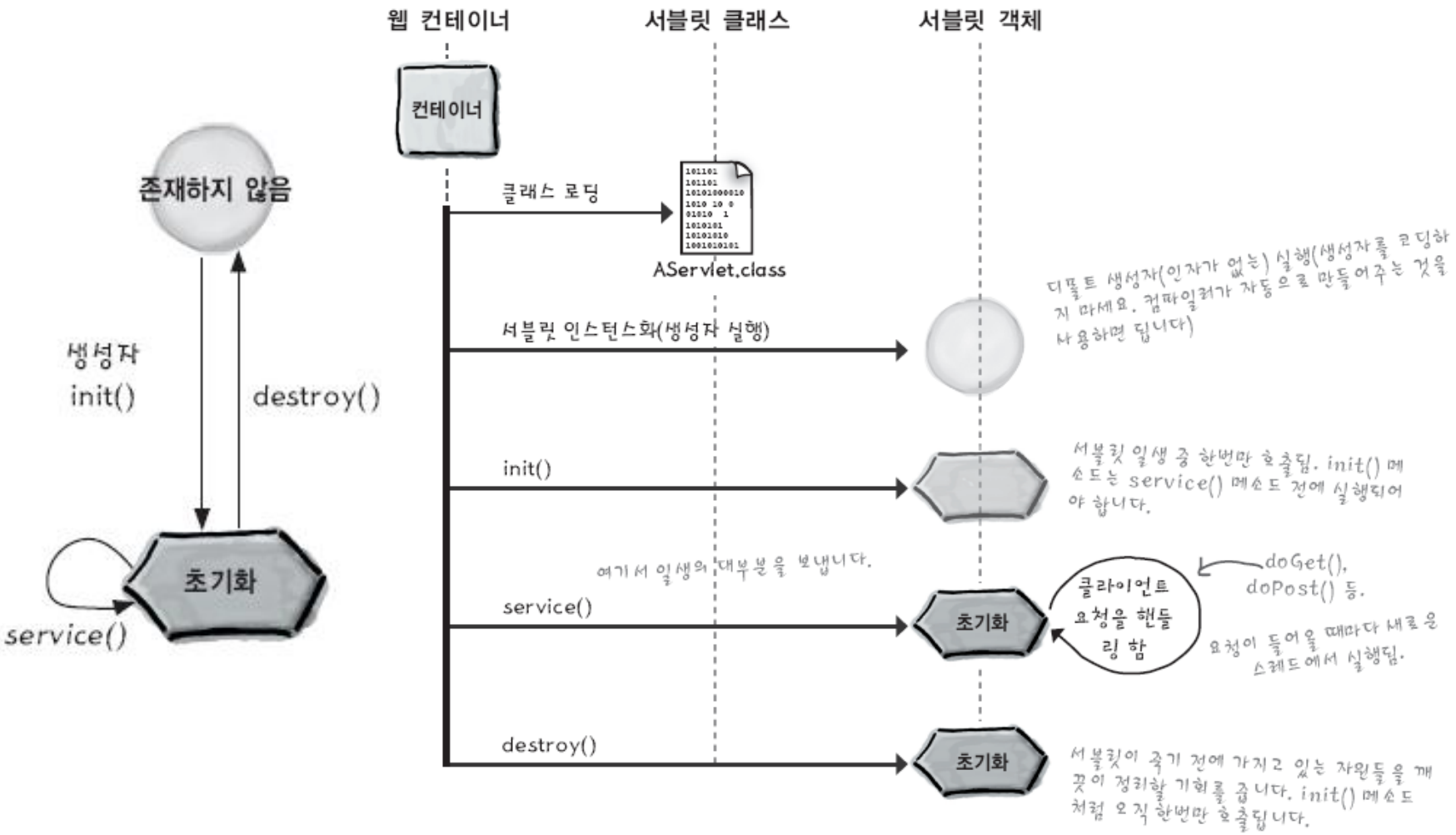
⑥



service() 메소드가 끝나면, 스레드를 소멸하거나 아니면 컨테이너가 관리하는 스레드 풀(Pool)로 돌려 보냅니다. 그 다음 Request와 Response 객체는 가비지 컬렉션이 될 준비를 할 것이며, 이 객체에 대한 참조는 이제 범위를 벗어나기에 사라집니다.

마지막으로 클라이언트는 서버로부터 응답을 받게 됩니다.

Servlet 라이프 사이클



Servlet 라이프 사이클 주요 메소드

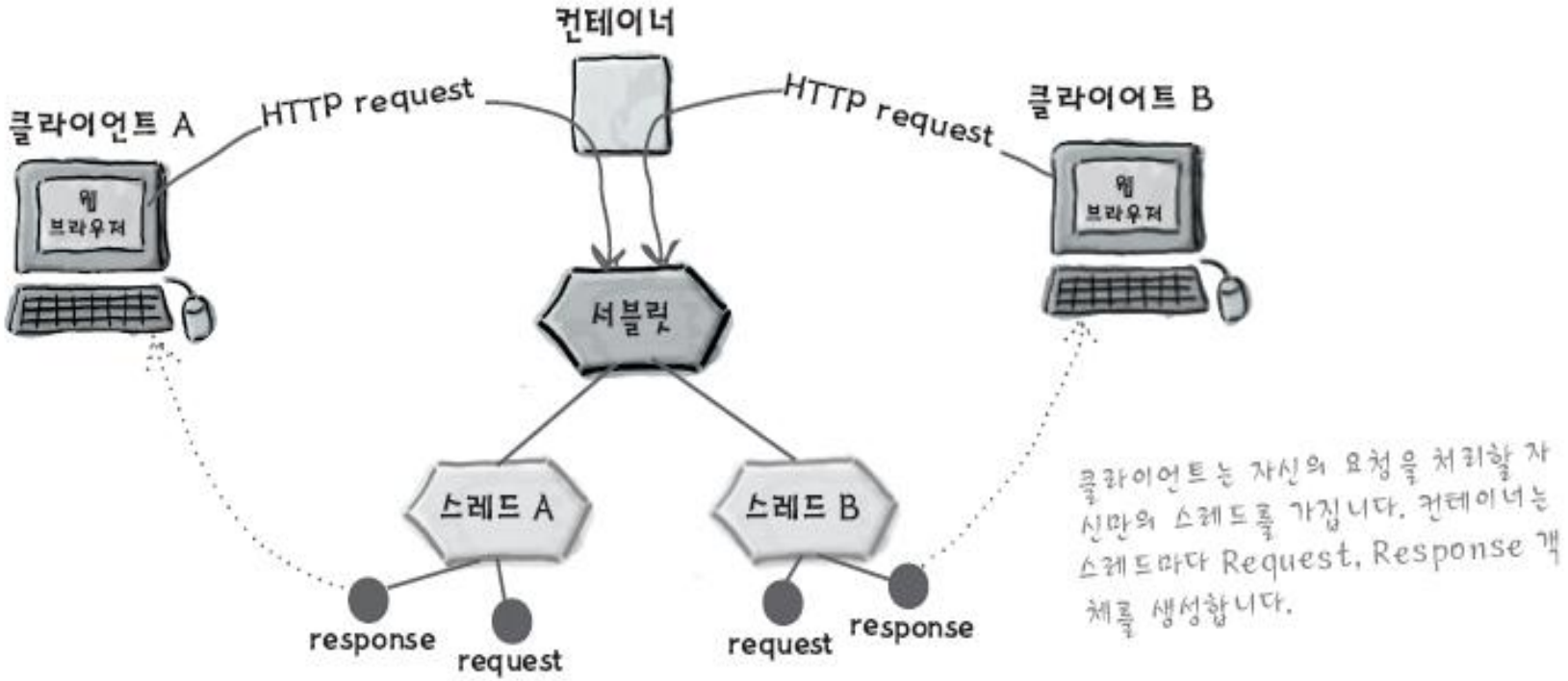
- init()
  - 컨테이너 에서 서블릿 객체를 생성한 다음에 호출한다. service() 이전에 실행
  - 서블릿을 초기화
  - 초기화할 내용(DB 접속 등)이 있는 경우 재정의
- service()
  - 클라이언트의 요청 후 컨테이너에서 쓰레드를 이용하여 호출
  - 요청의 HTTP 메소드(GET, POST등)를 참조하여 해당 메소드(doGet(), doPost() 등) 호출 판단
  - 거의 재정의 하지 않음
- destroy()
  - 서블릿 객체가 메모리에서 unload되기 직전에 호출
  - 소멸자와 유사한 역할

Servlet 라이프 사이클 주요 메소드

- doGet(), doPost()
  - service() 메소드에서 HTTP 메소드(GET, POST)를 참조하여 호출
  - 비즈니스 로직을 구현 또는 호출
  - 두 메소드 중 하나는 반드시 재정의하여 구현해야 한다.

Servlet 구동 방식

- 클라이언트의 요청은 서로 다른 쓰레드에서 실행

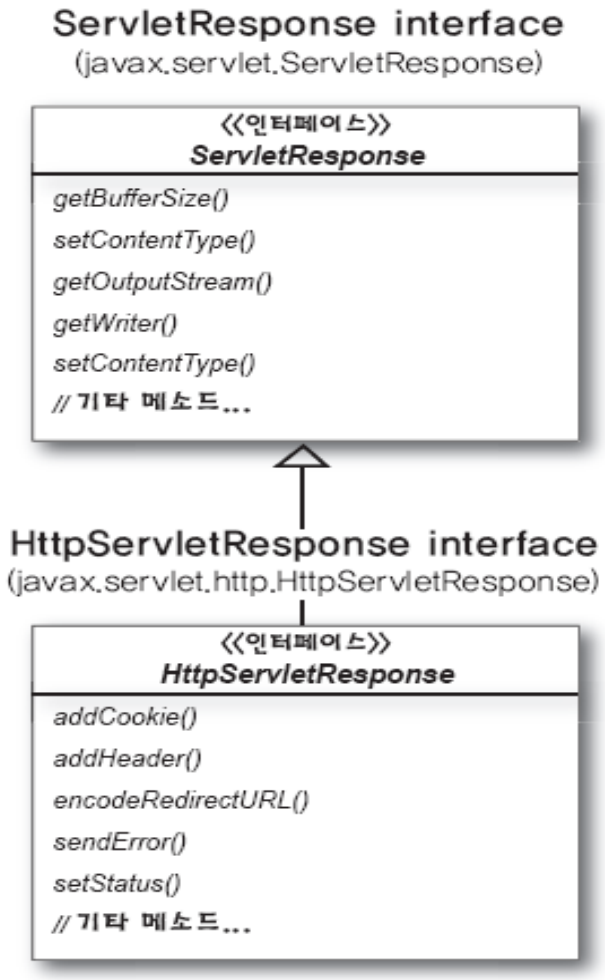


Servlet API

- HttpServletRequest
  - 쿠키, 헤더, 세션 등 HTTP 에 대한 것들에 대한 처리 관련 메소드 포함
  - HTTP 프로토콜에 관련된 메소드들이 추가되어 있음

Servlet API

- HttpServletResponse
  - HTTP에 관련된 오류, 쿠키, 헤더 정보에 대한 처리 관련 메소드 포함



Parameter 처리

- 요청 방식 설정
  - form태그의 method속성을 이용한다.
  - method속성 생략 시 GET방식

GET

단순한 하이퍼링크는 항상 GET을 의미합니다.

```
<A HREF="http://www.wickedlysmart.com/index.html/">click here</A>
```

POST

method="POST"라고 못박아 버리면, 당연히 POST죠.

```
<form method="POST"  action="SelectBeer.do">
  Select beer characteristics<p>
  <select name="color" size="1">
    <option>light
    <option>amber
    <option>brown
    <option>dark
  </select>
  <center>
    <input type="SUBMIT">
  </center>
</form>
```

Submit 버튼을 클릭하면 POST 요청의 몸체에 파라미터를 추가하여 전송합니다.

여기서는 color라는 파라미터가 되겠네요. 값은 <option> 항목에 있는 light, amber, brown, dark 중 사용자가 선택한 것이 되겠지요.

Parameter 처리

- 파라미터 이름 설정
  - input 요소의 name속성을 이용한다.

HTML 폼

```
<form method="POST"   action="SelectBeer.do">
  Select beer characteristics<p>
  <select name="color" size="1">
    <option>light
    <option>amber
    <option>brown
    <option>dark
  </select>
  <center>
    <input type="SUBMIT">
  </center>
</form>
```

요청 몸체에 네 개의 옵션 값 중 사용자가 선택한 값 하나를 color라는 이름의 파라미터로 전송합니다. 예를 들면, color=amber와 같이 말입니다.

Parameter 처리

HTTP POST 요청

```
POST /advisor/SelectBeer.do HTTP/1.1
Host: www.wickedlysmart.com
User-Agent: Mozilla/5.0 (Macintosh; U; PPC Mac OS X Mach-O; en-US; rv:1.4) Gecko/20030624
Netscape/7.1
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/
plain;q=0.8,video/x-mng,image/png,image/jpeg,image/gif;q=0.2,*/*;q=0.1
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive

color=dark
```

옆에 코드는 개발자가 작성하는 부분이 아니라, 브라우저가 자동으로 생성하는 것입니다. 단지 서버로 넘어가는 데이터가 이런 내용이다 정도는 봐 두어야 합니다.

서블릿 클래스

```
public void doPost(HttpServletRequest request, HttpServletResponse response)
                                                    throws IOException, ServletException {
    String colorParam = request.getParameter("color");
      // 여기에는 좀더 화려한 코드를 집어넣으면 되겠죠...
}
```

여기서 colorParam 문자 변수에는 dark라는 값이 들어 가겠죠.

폼에 있는 이름과 짝이 맞아야 합니다.

Parameter 처리

- 여러 개의 값을 가지는 파라미터의 경우
  - getParameterValues()를 이용하여 배열을 리턴받아야 한다.

Content 타입

- 응답내용의 데이터 타입을 정의한다.
- HTTP헤더 정보중의 하나이다.
- 종류
  - text/html
  - application/pdf
  - video/quicktime
  - image/jpeg
  - application/octet-stream
  - application/x-zip