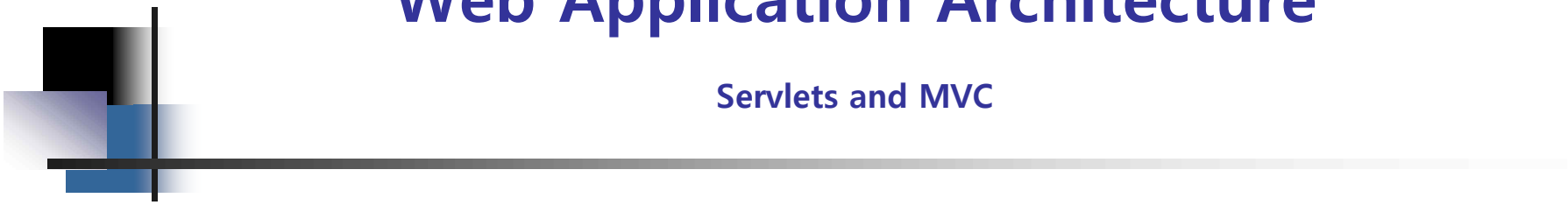


Web Application Architecture

Servlets and MVC

남 광 우

Servlets의 개요

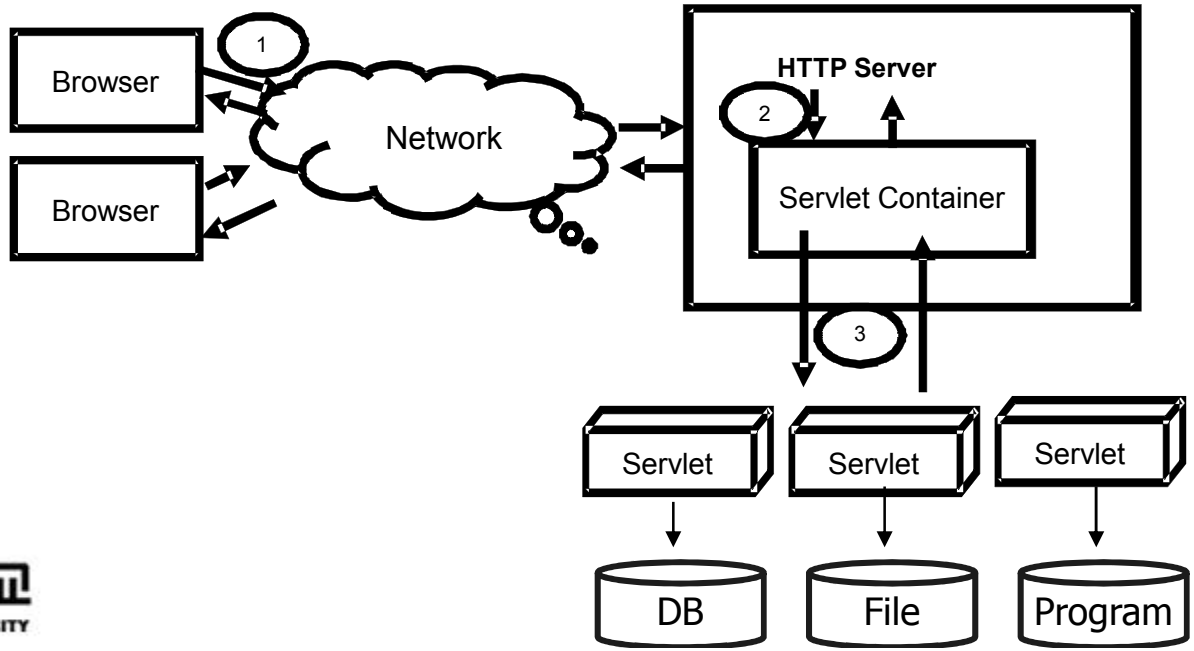
- Servlets 이란
  - 자바로 작성한 CGI 프로그램
  - Thread 를 이용
  - 플랫폼과 서버에 독립적
  - 확장성
  - 통합 개발 환경

- CGI와 Servlets의 비교

CGI	Servlets
Process 단위	Thread 단위
Process 생성시 자원 소모 , 속도 지연	최초 연결시에만 프로세스 생성(자원 절약, 속도 향상)
서버 부하	서버 효율성 유지

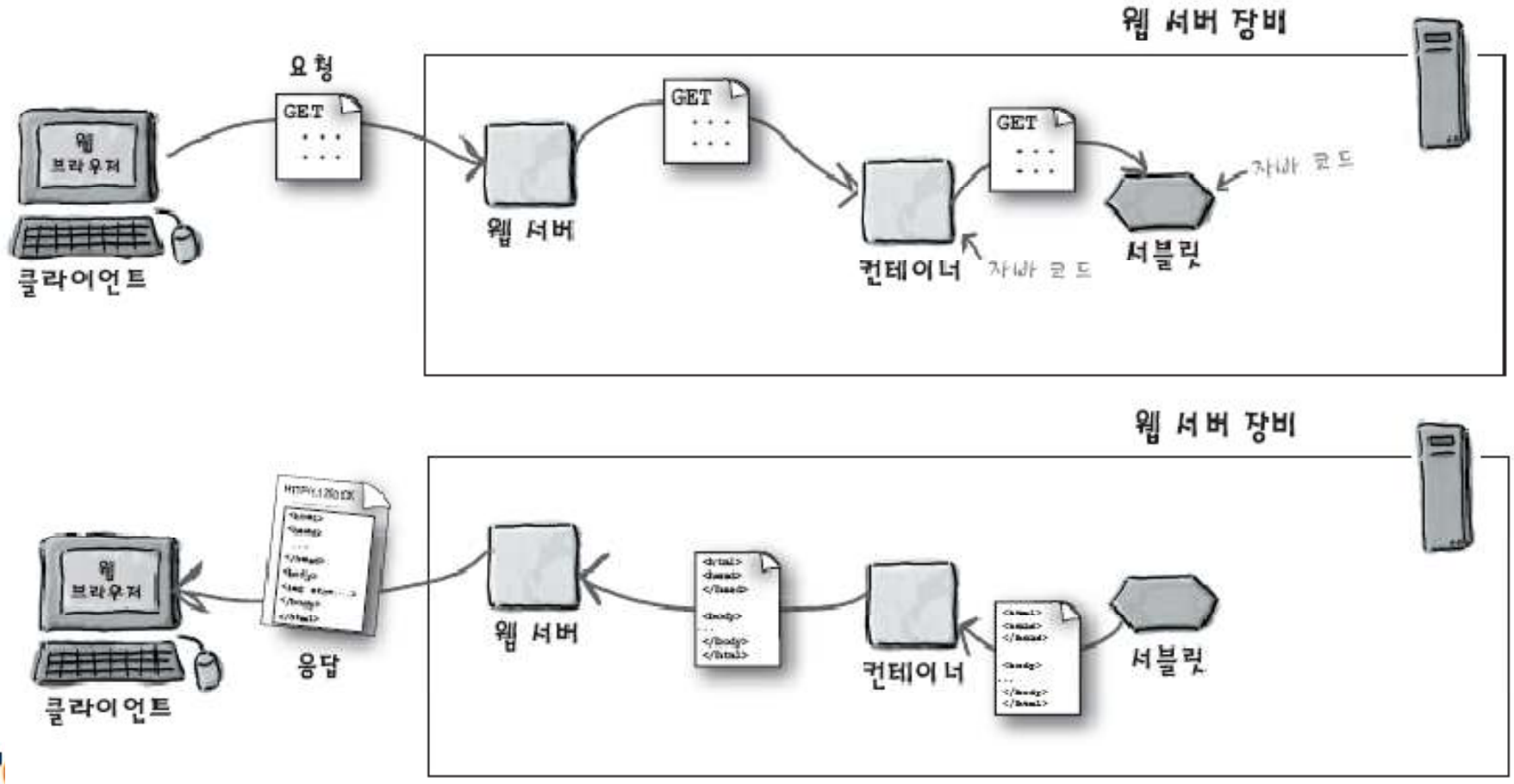
Servlets의 개요

- Servlets의 동작
  - 클라이언트의 요청
  - 서블릿 Handler 8080포트(변경가능)에서 요청 받음
  - 서블릿 컨테이너에서 해당 서블릿 검색
  - 해당 서블릿이 데이터 베이스 작업을 해야 한다면 DB연결, 작업
  - 모든 작업이 완료 되었다면 응답으로 결과를 리턴



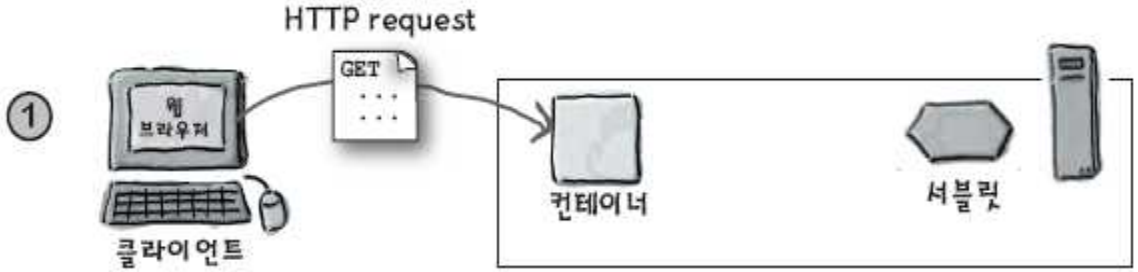
Servlets의 개요

- 서블릿 컨테이너의 정의 및 역할
  - HTTP 요청에 의한 서블릿을 실행시키며 관리하는 프로그램.

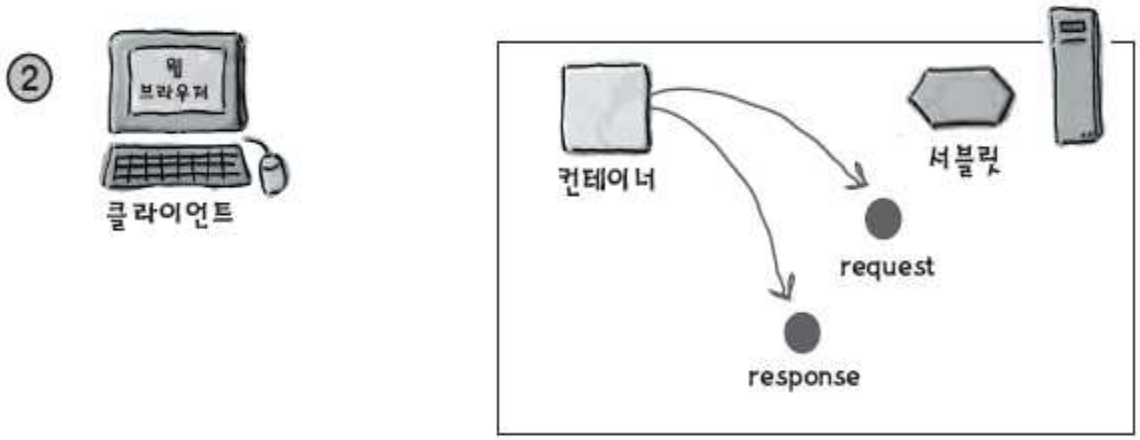


Servlets의 실행 과정

- Servlet에서 HTTP Request 처리 과정



① 사용자가 서블릿에 대한 링크를 클릭합니다.



② 컨테이너는 들어온 요청이 서블릿이라는 것을 간파하고 다음 두 객체를 생성합니다.

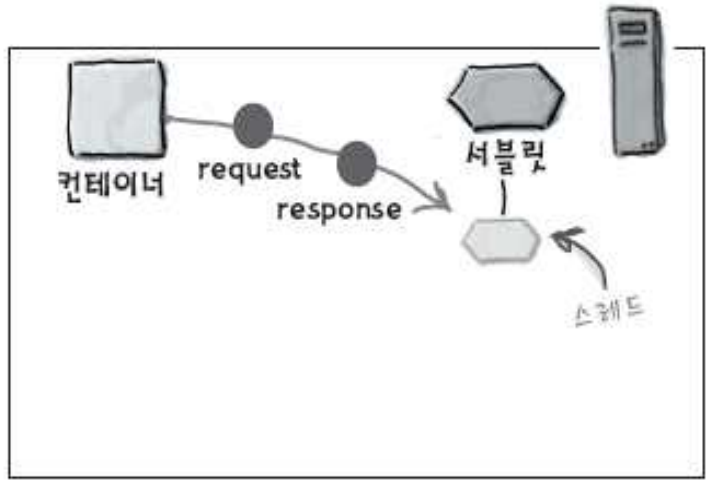
1) HttpServletResponse

2) HttpServletRequest

Servlets의 실행 과정

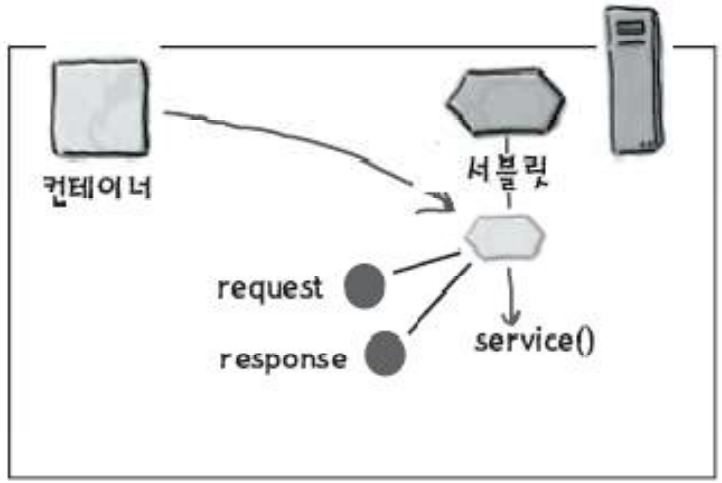
- Servlet에서 HTTP Request 처리 과정

③



사용자가 날린 URL을 분석하여 어떤 서블릿에 대한 요청인지 알아냅니다(여기서 DD를 참조합니다, 이해가 안되어도 그냥 넘어가기 바람). 그 다음 해당 서블릿 스레드를 생성하여 Request/Response 객체를 인자로 넘깁니다.

④



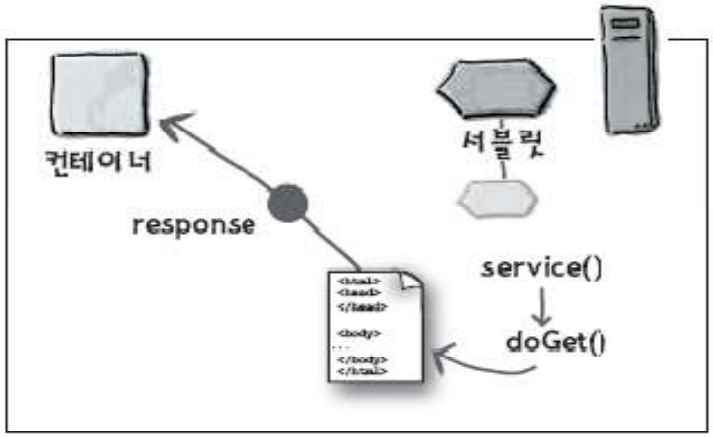
컨테이너는 서블릿 service() 메소드를 호출합니다. 요청에 지정한 방식(Method)에 따라 doGet()을 호출할지, doPost()를 호출할지 결정합니다.

여기서는 일단 HTTP GET 방식이라고 가정합시다.

Servlets의 실행 과정

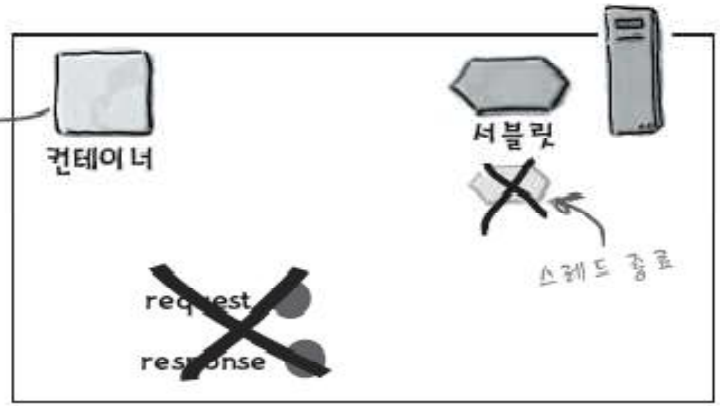
- Servlet에서 HTTP Request 처리 과정

⑤



doGet() 메소드는 동적인 페이지를 생성한 다음, 이를 Response 객체에 실어 보냅니다. 보내고 난 후에도 컨테이너는 Response 객체에 대한 참조(레퍼런스)를 가지고 있다는 것 잊지마세요.

⑥



스레드 작업이 끝나면, 컨테이너는 Response 객체를 HTTP Response로 전환하여 클라이언트로 내려 보냅니다. 이제 마지막으로 Request와 Response 객체를 소멸시킵니다.

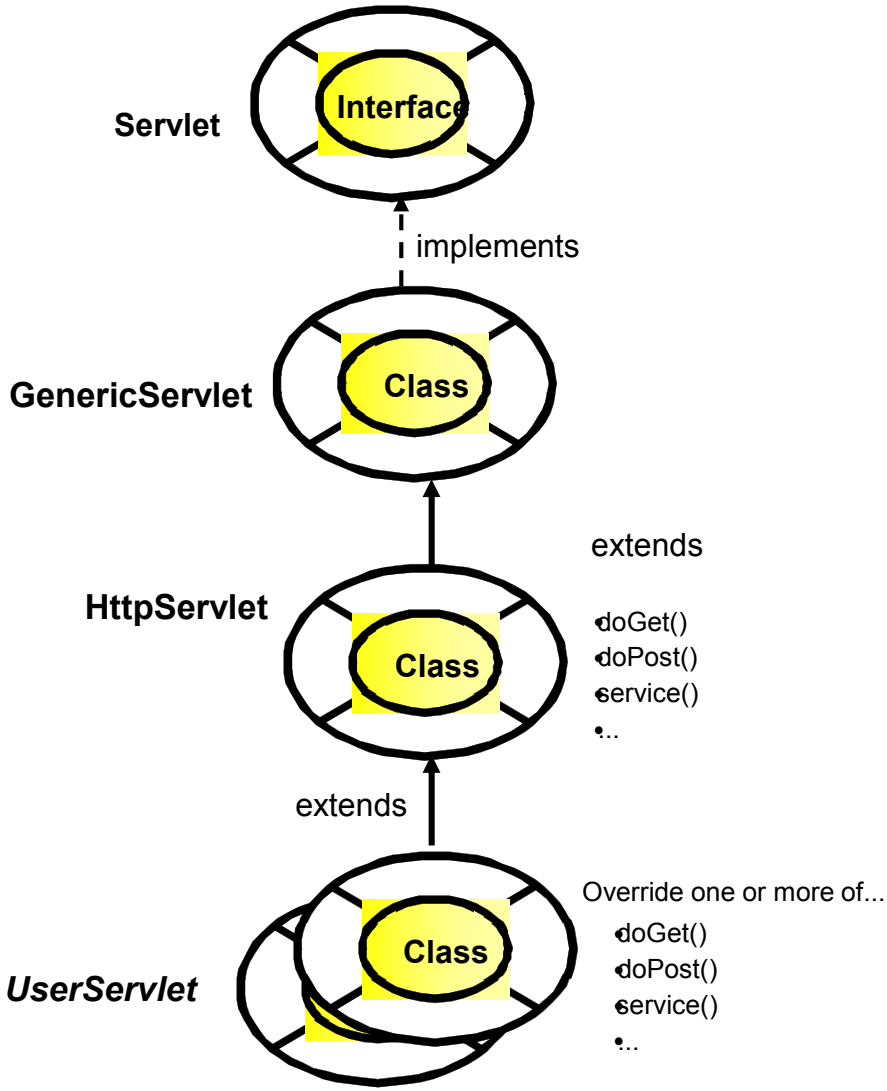
Servlets의 구현 구조

- Servlets의 주요 패키지
  - javax.servlet
    - 기본 servlets 정의를 포함
      - e.g. What is a servlet what are the inputs and outputs ...
    - 특정 프로토콜에 한정되지 않고 지원 (e.g., HTTP)
    - 저수준 classes/interfaces
    - 주요 클래스
      - GenericServlet
  - javax.servlet.http
    - HTTP 관련 정의 및 HTTP 프로토콜을 다루기 위한 기본 인터페이스 확장을 포함
    - 대부분의 웹 프로그래밍에서 HTTP 관련 클래스를 사용
    - 주요 클래스
      - HttpServlet

Servlets의 구현 구조

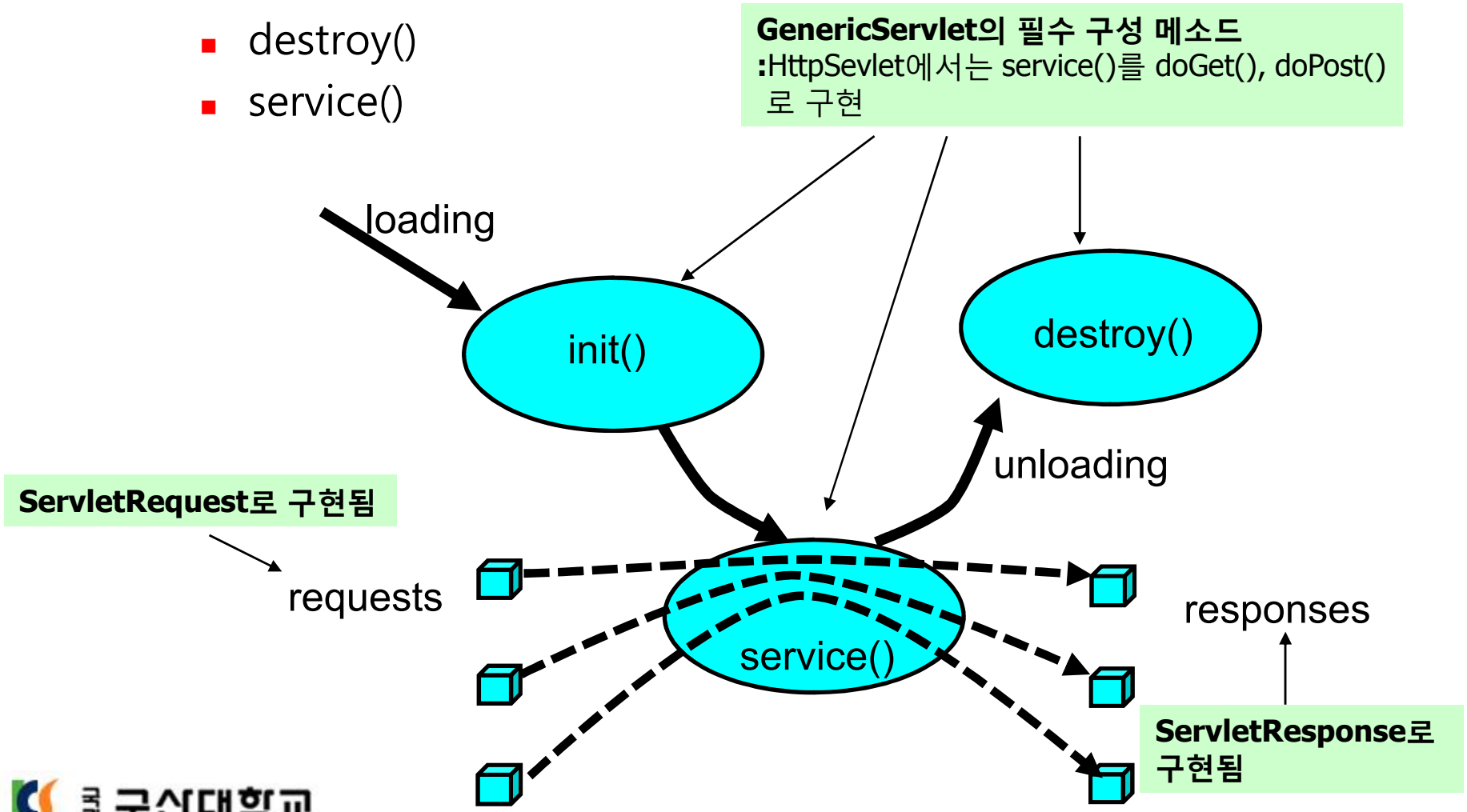
- javax.servlet.http
  - Interfaces
    - HttpServletRequest
    - HttpServletResponse
    - HttpSession
  - Classes
    - HttpServlet
    - Cookie

Servlets의 메인 클래스



Servlets의 구현 구조

- Servlets의 라이프 사이클
  - init()
  - destroy()
  - service()

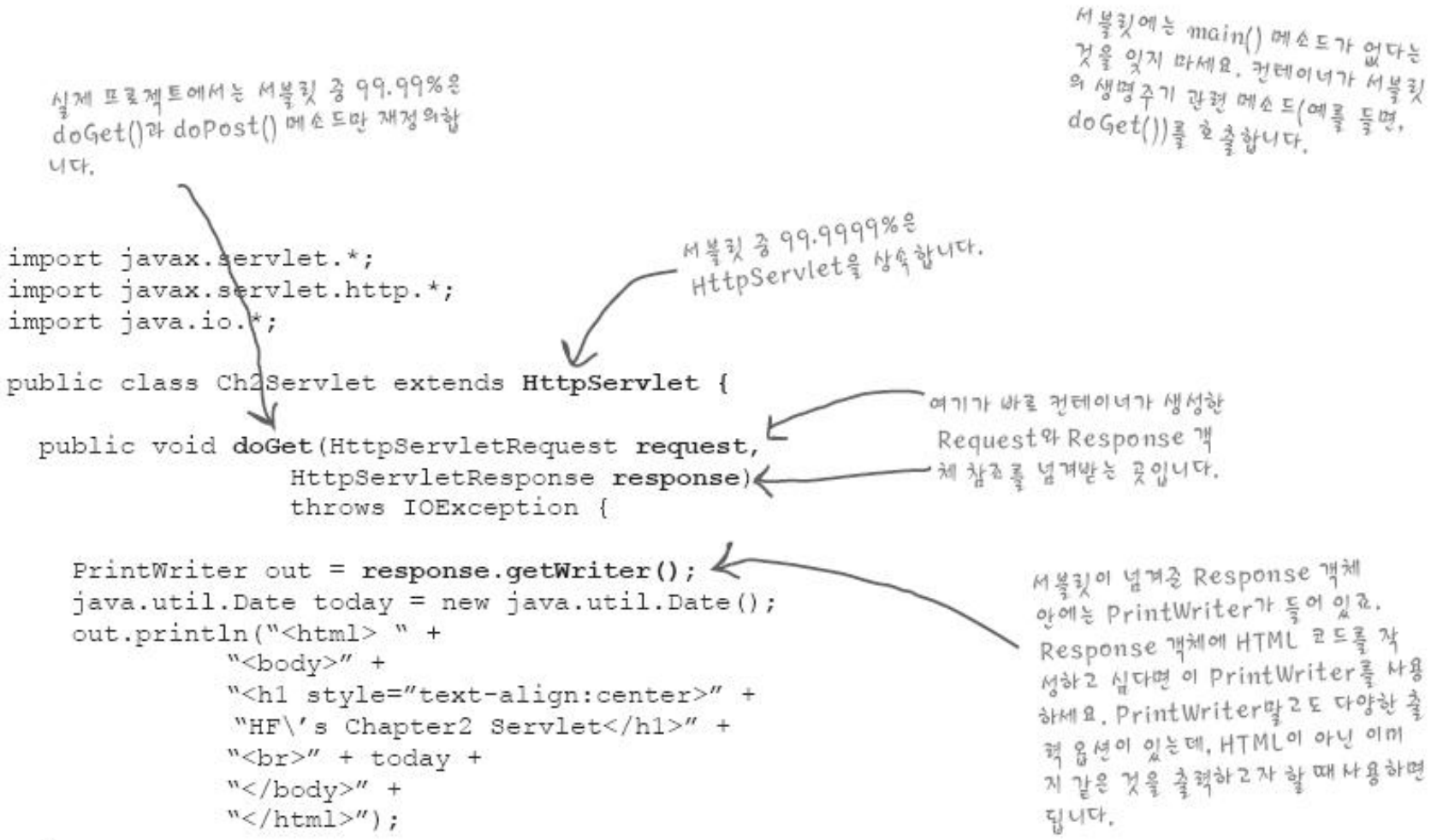


Servlets의 구현 구조

- Servlets 인터페이스의 주요 메소드
  - void init(ServletConfig config)
    - 서블릿을 시작할 때 호출하여, 필요한 자원을 할당하는 등의 서블릿을 초기화 하고, 서비스를 시작할 수 있도록 함
  - void service(ServletRequest req, ServletResponse res)
    - 서블릿이 초기화 된 후 클라이언트로부터 온 요청을 처리
  - void destroy()
    - 서블릿을 종료할 때 호출하여, 필요한 자원을 할당 해제
  - ServletConfig getServletConfig()
    - 초기화/시작 매개변수 등을 포함하고 있는 ServletConfig 객체 반환
  - java.lang.String getServletInfo()
    - 서블릿의 작성자(author), 버전(version), 저작권(copyright) 등과 같은 서블릿 관련 정보를 제공

Servlets의 구현 구조

- 간단한 Servlet의 예




```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class Ch2Servlet extends HttpServlet {

  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
                    throws IOException {

    PrintWriter out = response.getWriter();
    java.util.Date today = new java.util.Date();
    out.println("<html> " +
                "<body>" +
                "<h1 style="text-align:center>" +
                "HF\'s Chapter2 Servlet</h1>" +
                "<br>" + today +
                "</body>" +
                "</html>");
  }
}
```

Servlets의 구현 구조

- Servlet의 배포
  - 실제 Servlet 구현과 서비스를 위한 URL을 독립적이 되도록 함

클라이언트가 아는 *URL* 이름

클라이언트는 사실 HTML 안에 존재하는 서블릿 이름만 알고 있으면 됩니다. 서버 상의 실제 어느 디렉토리에 어떤 파일명으로 존재하는지는 관심 없죠. 사실 URL 이름은 클라이언트를 위한 애칭이라고 표현해도 되겠네요.

배포자가 만든 *내부적인* 이름

배포자는 실제 애플리케이션 운영을 위하여 배포명이란 것을 만들어 이를 개발자에게 알려줍니다. 배포명도 사실 URL 이름과 같이 가공의 이름이라고 말할 수 있습니다. 이 이름은 URL 이름과 같을 필요가 없으며, 실제 서블릿 파일 위치 이름과 일치하지 않아도 됩니다.

실제 *파일*명

개발자가 만든 서블릿 클래스 안에는 클래스명과 패키지명이 들어 있습니다. 마찬가지로 서블릿 클래스 파일도 또한 파일 시스템 상의 경로와 파일명이 있습니다. 이는 패키지 디렉토리 구조에 따라 서버마다 다를 수 있습니다.

Servlets의 구현 구조

- Servlet Deployment를 위한 기본 요소

  - 배포 서술자 (DD, Deployment Descriptor)
    - 서블릿 컨테이너에 서블릿 배포 시 사용하는 XML 문서
    - URL과 서블릿 매핑 정보 포함
    - 보안역할 설정, 오류 페이지 설정, 초기화 구성 및 관련 정보 설정 등

  - DD 내의 주요 element
    - <servlet>
      - 서블릿 내부명과 완전한 클래스명과의 매핑정보
    - <servlet-mapping>
      - 서블릿 내부명과 URL 명과의 매핑정보

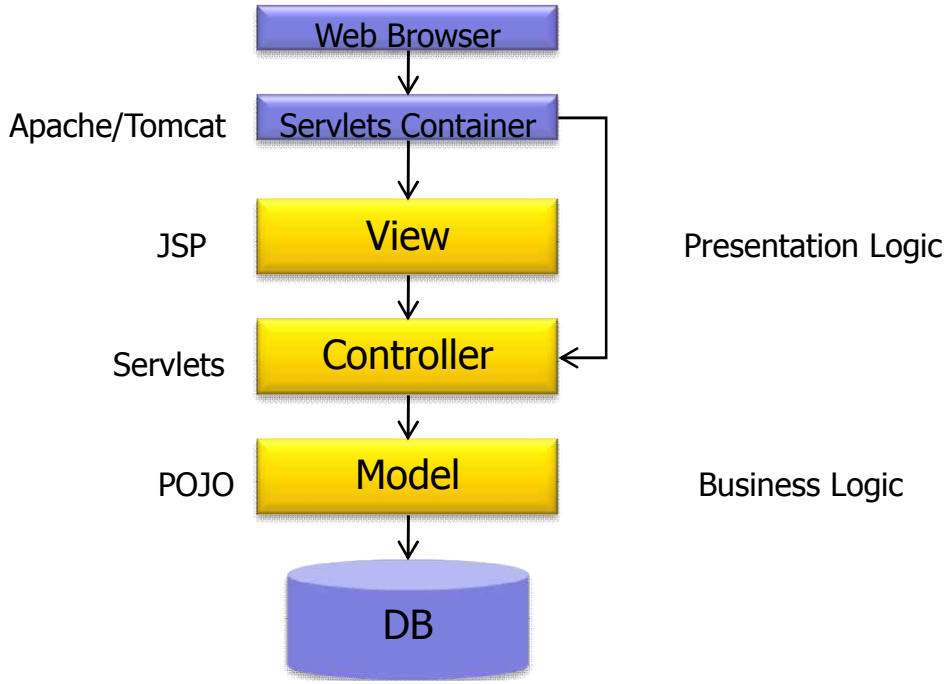
  - WEB-INF/web.xml 파일에 기술

Servlets의 구현 구조

- Web.xml 파일의 예

MVC 디자인 패턴

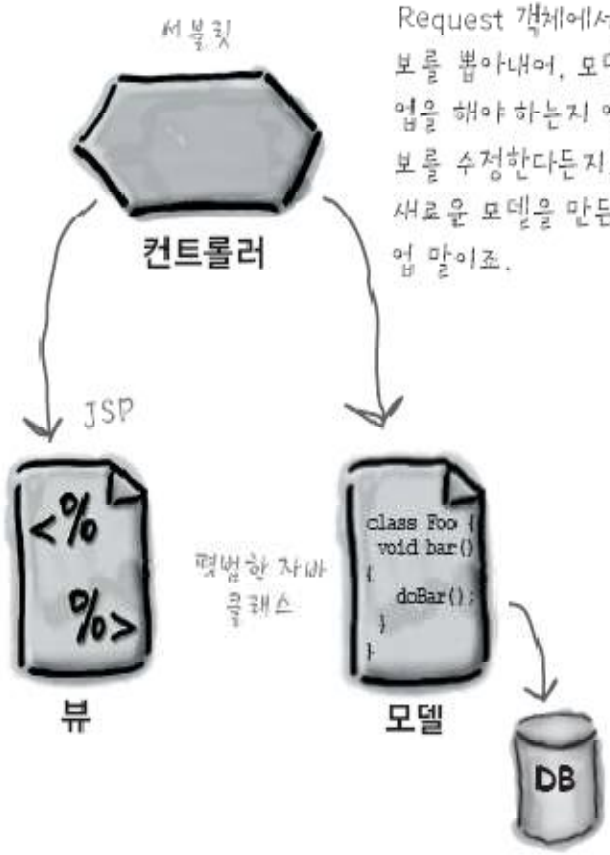
- MVC 란?
  - 모델 (Model) – 컨트롤러 (Controller) – 뷰 (View)의 약자
  - 비즈니스 로직과 프리젠테이션 로직의 분리를 위해 사용
    - 비즈니스 로직의 재사용 :
    - 프리젠테이션 로직의 변경에 영향을 받지 않음
    - 서블릿과 비즈니스 로직의 분리가 필요



MVC 디자인 패턴

- MVC의 보다 자세한 설명

서블릿, JSP 환경에서 MVC



컨트롤러(Controller)

Request 객체에서 사용자가 입력한 정보를 뽑아내어, 모델에 대하여 어떤 작업을 해야 하는지 알아냅니다. 모델 정보를 수정한다든지, 뷰(JSP)에게 넘겨줄 새로운 모델을 만든다든지 등과 같은 작업 말이죠.

뷰(View)

프리젠테이션에 대한 책임을 지죠. 뷰는 컨트롤러로부터 모델 정보를 읽어옵니다(직, 간접적인 방법 둘 다 가능하며, 뷰가 칠을 수 있는 곳에 컨트롤러가 갖다 두는 방식을 많이 사용합니다). 뷰는 또한 사용자가 입력한 정보를 컨트롤러에게 넘겨주어야 합니다

모델(Model)

비즈니스 로직이 바로 여기에 들어갑니다. 모델 정보(state)를 읽어오거나(getter) 수정하는(setter) 로직도 여기 포함됩니다.

장바구니 예를 들면, 장바구니에 들어 있는 상품이 바로 모델입니다(여기엔 이를 어떻게 처리한다라는 내용도 포함되겠죠).

MVC 패턴에서 모델은 데이터베이스와 통신하는 유일한 곳입니다(물론 DB 통신만을 전담하는 객체를 따로 빼낼 수도 있지만... 이 패턴은 뒤에서 다루겠습니다).

Servlets 예제 - HelloWorld

- 예 : Hello Servlet!

HelloWorldServlet.java

```
import java.io.*;

import javax.servlet.*;
import javax.servlet.http.*;

public class HelloWorldServlet extends HttpServlet {

 public void init() throws ServletException {
 //  Initialize and run when loaded.  Can use default.
 }
 public void destroy() {
 // Release resources, exit, etc. Can use default.
 }
 public void doPost(HttpServletRequest req, HttpServletResponse res) {
    doGet( req, res );
}
 public void doGet(HttpServletRequest req, HttpServletResponse res)
   throws ServletException, IOException
 {
                res.setContentType("text/html");

                PrintWriter out = res.getWriter();
                out.println("<html>");
                out.println("<head><title>Hello World</title></head>");
                out.println("<body>");
                out.println("<h1>Hello World</h1>");
                out.println("</body></html>");
                out.close();
 }
}
```

Servlets 예제 - HelloWorld

- 예 : HelloWorld의 호출 방법
  - servlet's URL 직접 호출
    - http://<yourServer>/servlet/ServletToCall
  - HTML <form> tag을 이용한 호출
    - <form method=GET action=/servlet/ServletToCall> ... </form>
    - <form method=POST action=/servlet/ServletToCall> ... </form>
  - HTML <servlet> </servlet> tags를 이용한 호출
    - <servlet name=ServletToCall code=ServletToCall.class>
    - ..
    - </servlet>

Servlets 예제 - HelloWorld

- 예 : Servlet의 컴파일
  - Eclipse WTP를 이용한 방법
    - J2EE 수준의 개발 환경 지원
      - deployment까지 지원하나 다소 무거움
    - 다음 URL을 참조
      - http://wiki.javajigi.net/pages/viewpage.action?pageId=163
  - sysdeo plug-in
    - tomcat의 startup-shutdown을 지원
    - 간단하면서 가벼움
    - 다음의 URL을 참조
      - http://www.ibm.com/developerworks/kr/library/os-ectom/
  - javax.servlet 라이브러리 문제
    - tomcat/lib 아래의 servlet-api.jar를 build-path에 추가

Servlets 예제 - HelloWorld

- 예 : HelloWorld의 deployment
  - Tomcat의 디렉토리 구조

web.xml
servlet 클래스 설정 파일

Tomcat서버 실행파일

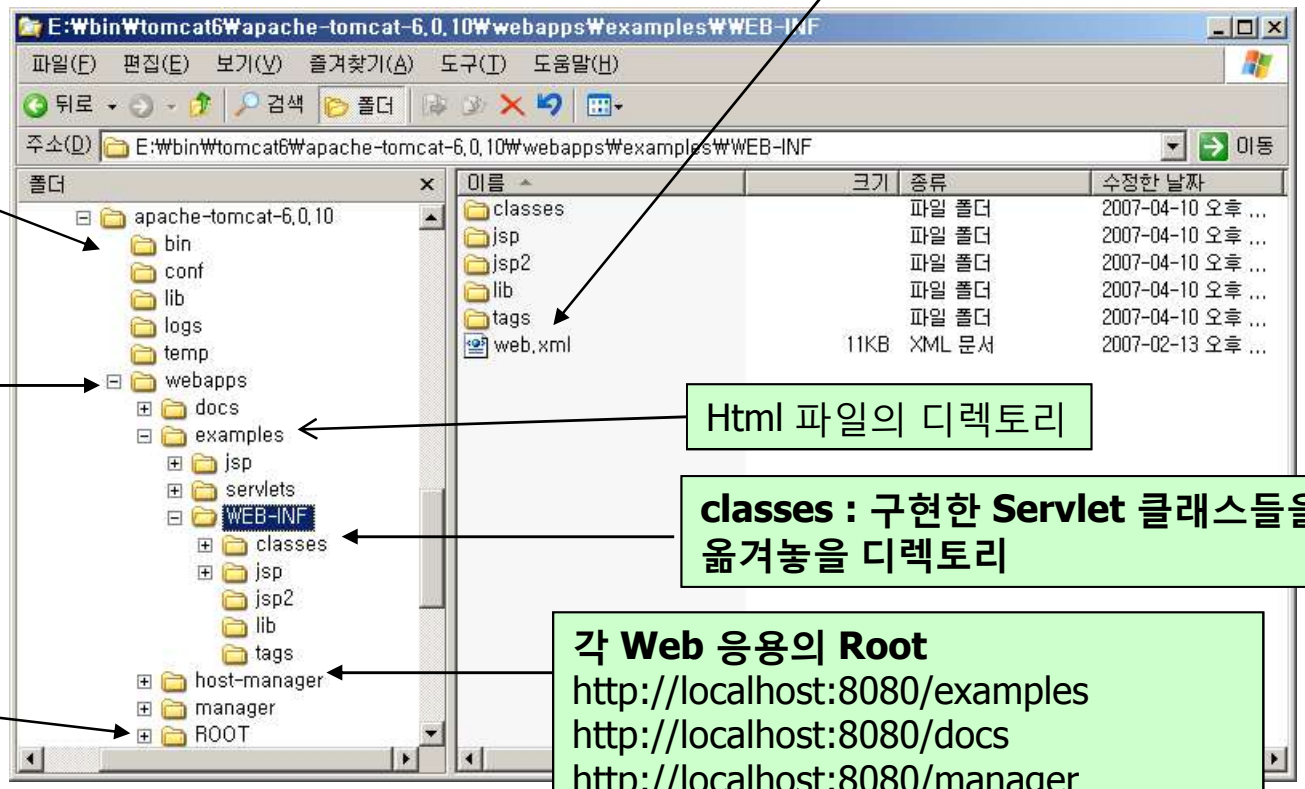
Web 응용 디렉토리

Html 파일의 디렉토리

classes : 구현한 Servlet 클래스들을 옮겨놓을 디렉토리

Tomcat의 웹 Root
http://localhost:8080/

각 Web 응용의 Root
http://localhost:8080/examples
http://localhost:8080/docs
http://localhost:8080/manager



Servlets 예제 - HelloWorld

- 예 : HelloWorld의 deployment
  - web.xml의 설정
    - servlet 구현 클래스를 webapps/ROOT/WEB-INF/classes에 카피
    - webapps/ROOT/WEB-INF/ 디렉토리에 web.xml을 작성
    - http://localhost:8080/hello를 이용하여 호출

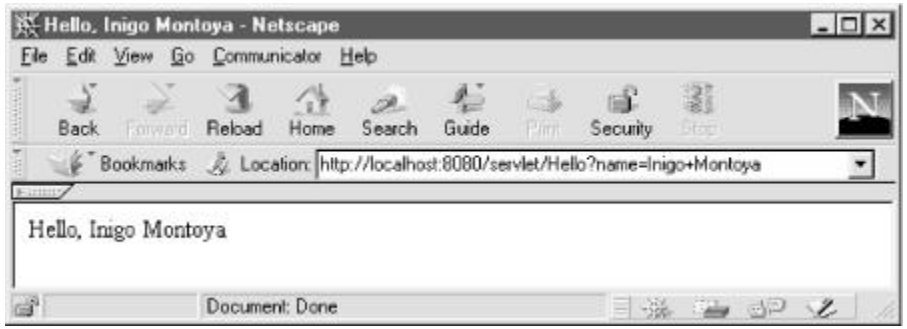
web.xml의 예

```
<!DOCTYPE web-app PUBLIC '-//Sun Microsystems, Inc.//DTD Web Application
2.3//EN' 'http://java.sun.com/dtd/web-app_2_3.dtd'>
<web-app>
 <servlet>
   <servlet-name>hello</servlet-name>
   <servlet-class>HelloWorldServlet</servlet-class>
 </servlet>

 <servlet-mapping>
   <servlet-name>hello</servlet-name>
   <url-pattern>/hello</url-pattern>
 </servlet-mapping>
</web-app>
```

Servlets 예제 – Hello You!

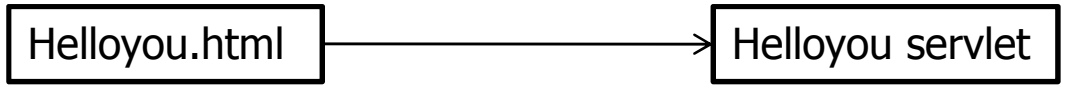
- 예2: 웹에서 이름 입력받아 Hello!
  - 다음 그림과 같이 웹에서 이름을 입력받아 화면에 출력하는 프로그램을 작성하시오.

자기 이름을 입력하고
버튼을 Click



Servlets 예제 – Hello You!

- 예2: 웹에서 이름 입력받아 Hello!의 설계



1) 이름을 입력하고 버튼을 클릭
2) Helloyou 서블릿을 호출

1) HttpServletRequest의 파라미터로 전송받은 이름을 추출
2) 이름과 Html을 복합한 내용을 HttpServletResponse를 통해 전송

Servlets 예제 – Hello You!

- 예2: 웹에서 이름 입력받아 Hello!의 구현

Helloyou.html

```
<HTML>
<HEAD>
<TITLE>Introductions</TITLE>
</HEAD>
<BODY>
  <FORM METHOD=GET ACTION="/HelloYou">
      If you don't mind me asking, what is your name?
    <INPUT TYPE=TEXT NAME="name"><P>
     <INPUT TYPE=SUBMIT>
  </FORM>
</BODY>
</HTML>
```

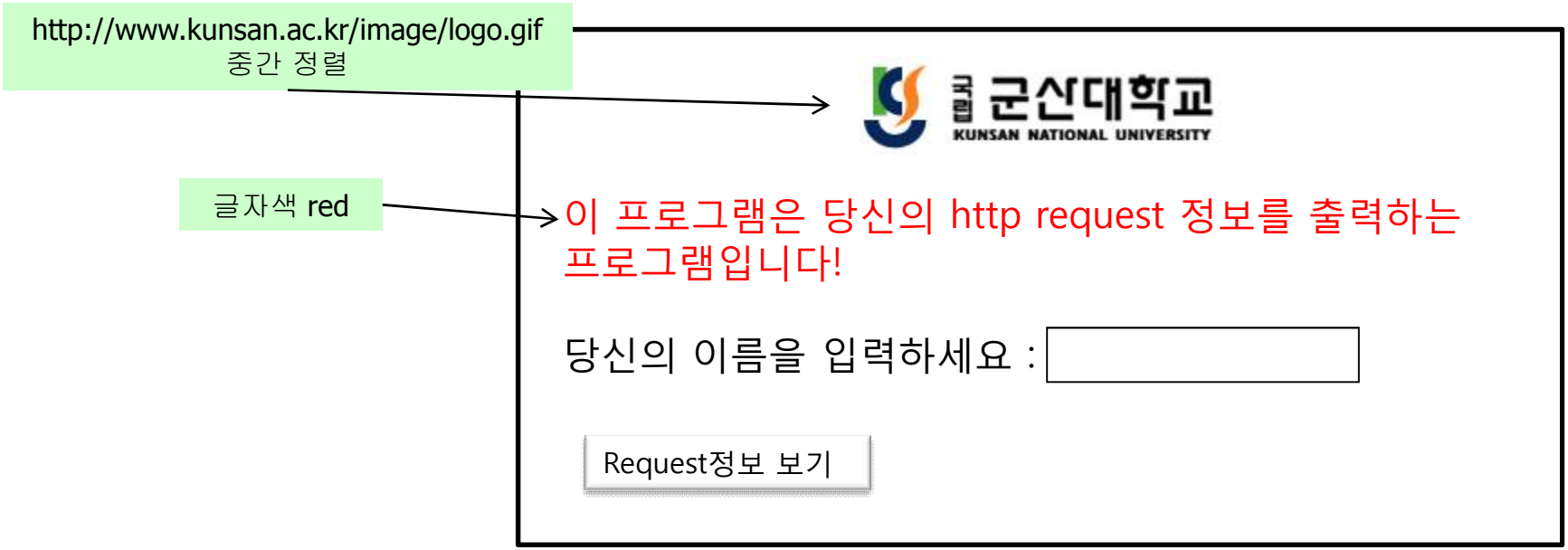
Servlets 예제 – Hello You!

- 예2: 웹에서 이름 입력받아 Hello!의 구현

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class Helloyou extends HttpServlet {

 public void doGet(HttpServletRequest req, HttpServletResponse res)
                        throws ServletException, IOException {

   res.setContentType("text/html");
   PrintWriter out = res.getWriter();

   String name = req.getParameter("name");
   out.println("<HTML>");
   out.println("<HEAD><TITLE>Hello, " + name + "</TITLE></HEAD>");
   out.println("<BODY>");
   out.println("Hello, " + name);
   out.println("</BODY></HTML>");
 }

 public String getServletInfo() {
   return "A servlet that knows the name of the person to whom it's" +
          "saying hello";
 }
}
```

Servlets 예제 - 실습

- 실습 : Request 정보 보기
  - 다음의 웹 화면과 같이 이름을 입력받아 Request 정보를 제공하는 프로그램을 작성하시오
    - 이 프로그램이 다음과 같이 실행될 수 있도록 설정 및 구현
      - 웹 응용의 이름과 위치 : $(TOMCAT)/wepapps/FirstServlet
      - HTML 파일의 접근 : http://localhost:8080/FirstServlet/firstReq.html
      - 서블릿 실행 : http://localhost:8080/FirstServlet/firstReq



Servlets 예제 - 실습

- 실습 : Request 정보 보기 Servlet

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class RequestInfo extends HttpServlet {

   public void doGet(HttpServletRequest request, HttpServletResponse response)
   throws IOException, ServletException
   {
      response.setContentType("text/html");
      PrintWriter out = response.getWriter();
      String name = req.getParameter("name");
      out.println("<html>");
      out.println("<body>");
      out.println("<head>");
      out.println("<title>KSNU Request Info : </title>");
      out.println("</head>");
      out.println("<body>");
      out.println("<h3>Request Information Example</h3>");
      out.println("<h3> Your name is :  "+ name + "</h3>");
      out.println("Your http request information is ....");
      out.println("Method: " + request.getMethod());
      out.println("Request URI: " + request.getRequestURI());
      out.println("Protocol: " + request.getProtocol());
      out.println("PathInfo: " + request.getPathInfo());
      out.println("Remote Address: " + request.getRemoteAddr());
      out.println("</body>");
      out.println("</html>");
   }


   public void doPost(HttpServletRequest request, HttpServletResponse response)
   throws IOException, ServletException
   {
      doGet(request, response);
   }
}
```

웹을 통해 입력 받은 이름