

jsp 2강 - 서블릿

양 명 속

[now4ever7@gmail.com]



목차

- 서블릿
- Servlet의 동작원리
- 간단한 서블릿 작성하기
- 웹 컨테이너
- 서블릿 라이프 사이클
- 서블릿 예제 - 상품 등록

Servlet의 동작원리

Servlet
↑
GenericServlet
↑
HttpServlet

- 서블릿 - 자바의 클래스 중 오직 서버에서만 해석되어 실행되어 지는 클래스, 서블릿 규약에 따라 만든 클래스
- 서블릿 생성
 - 서블릿 API 이용
 - javax.servlet, javax.servlet.http 패키지 이용

javax.servlet

- 프로토콜에 독립적인 서블릿을 만들기 위한 클래스 제공

javax.servlet.http

- HTTP 프로토콜의 고유한 기능(GET, POST 등)을 제공하는 서블릿을 만드는 클래스를 제공

- 서블릿 구현
 - GenericServlet 이나 **HttpServlet** 중 하나를 **상속** 받아서 서블릿 구현
 - main() 메서드를 갖지 않음
 - 대신, 서블릿의 특정한 메서드는 서버(웹 컨테이너)가 호출함
 - 서버가 서블릿에 요청을 전달할 때마다 서블릿의 **service()** 메소드가 호출됨

Servlet의 동작원리

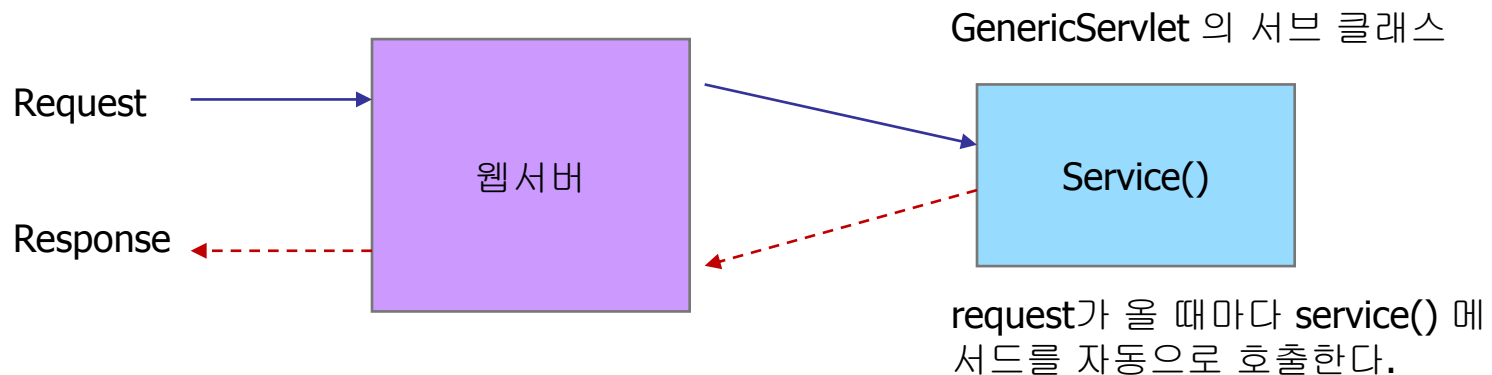
■ GenericServlet

- 요청을 처리하기 위해 자신의 **service()** 메서드를 오버라이드해야 함
- service() 메서드
 - 두 개의 매개변수(request객체, response객체)를 받아들임

■ HttpServlet

- service() 메서드를 오버라이드하지 않고, Get요청을 다루기 위해 **doGet()**을 오버라이드, POST 요청을 다루기 위해 **doPost()**를 오버라이드함

- javax.servlet.http 패키지의 HttpServletRequest와 HttpServletResponse는 HTTP 요청과 응답에 대한 기능을 제공함



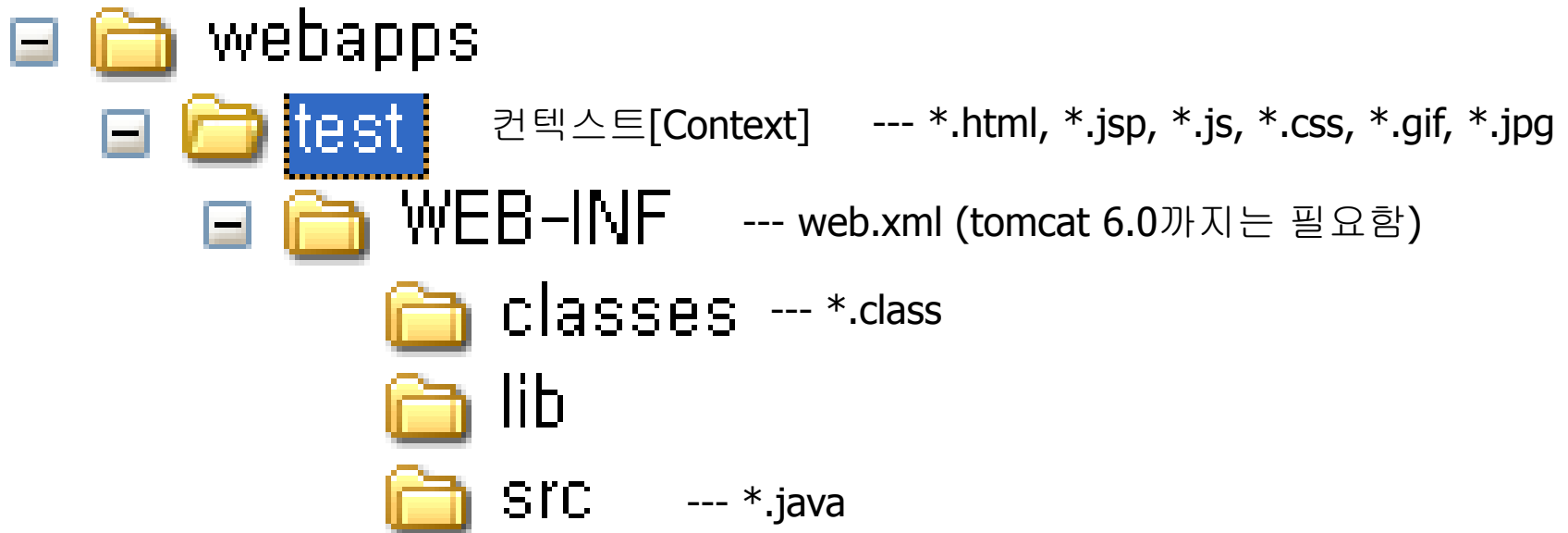
request 를 처리하는 GenericServlet



간단한 서블릿 작성하기

- 서블릿을 이용한 웹 어플리케이션 개발 과정
 - 서블릿 소스 코드를 저장할 디렉토리를 생성한다
 - 클래스 파일을 저장할 WEB-INF\classes 디렉토리 생성
 - CLASSPATH 환경 변수 값을 설정
 - 서블릿 소스 코드를 작성
 - 소스 코드를 컴파일한 뒤, 생성된 클래스 파일을 classes 디렉토리에 복사
 - WEB-INF\web.xml 파일에 서블릿 정보를 설정 (tomcat 6.0이하인 경우)
 - 웹 컨테이너를 시작한다
 - 웹 브라우저에서 테스트한다.
- 소스 코드 저장 디렉토리
 - test\WEB-INF\src
- 클래스 파일 저장 디렉토리
 - test\WEB-INF\classes
 - 웹 어플리케이션에서 필요로 하는 클래스를 웹 어플리케이션 디렉터리의 WEB-INF\classes 디렉토리에 위치시키도록 되어 있음

폴더 구조



※ 폴더 구조

d:\ Tomcat 8.0\ webapps\ test

test 폴더 - html, 이미지 파일, js, css, jsp 등이 위치함

test\ WEB-INF\ classes - 자바 클래스파일

test\ WEB-INF\ src - 자바 소스 파일

test\ WEB-INF\ lib - 라이브러리



간단한 서블릿 작성하기

- CLASSPATH 환경변수 - 소스코드를 컴파일하거나 자바 클래스를 실행할 때 참조되는 클래스의 경로를 담고 있는 환경변수
- servlet-api.jar
 - 서블릿을 개발하는 데 필요한 클래스 파일을 포함하고 있는 jar 파일

CLASSPATH - [톰캣 설치 디렉토리]\ lib\ servlet-api.jar

- 웹 어플리케이션에 서블릿 등록하기
 - 웹 어플리케이션에서 서블릿을 사용할 수 있도록 하려면 **web.xml 파일에 서블릿을 설정**해 주어야 함 (tomcat 6.0 이하인 경우)
 - web.xml - 웹 어플리케이션의 서블릿, 필터, 리스너 및 관련 매핑 정보를 담고 있는 설정 파일
 - 서블릿 규약은 서블릿 및 요청 URL을 어떤 서블릿이 처리할 것인지에 대한 매핑 정보를 WEB-INF\web.xml 파일에서 설정하도록 정의하고 있음



java EE API

- Java EE에 필요한 API
 - www.oracle.com 들어가서
왼쪽 메뉴에 java EE -> [Documentation] 탭 ->
 - <http://www.oracle.com/technetwork/java/javaee/documentation/index.html>
 - API Documentation
 - Java EE 7 클릭
=> <http://docs.oracle.com/javaee/7/api/>
- Java SE API
 - <http://docs.oracle.com/javase/8/docs/api/index.html>

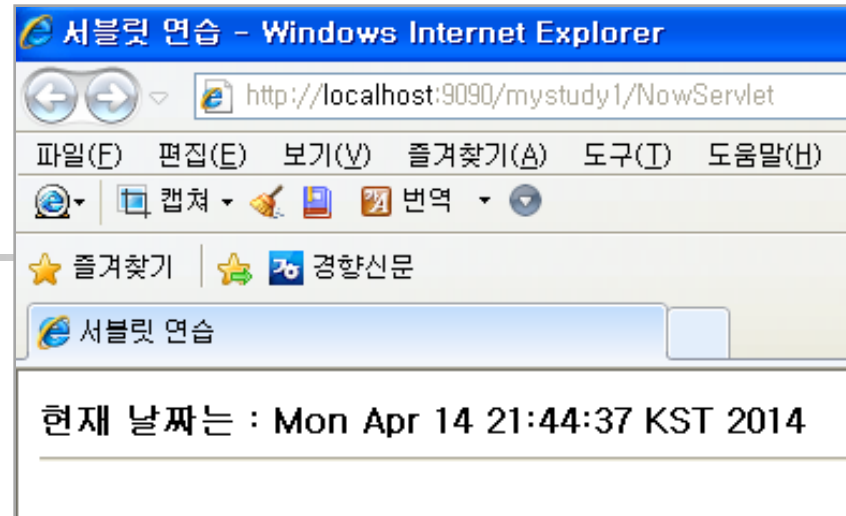
http://localhost:9090/test/NowServlet

예제

```
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Date;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.annotation.WebServlet;
```

```
@WebServlet("/NowServlet")
public class NowServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request,
                          HttpServletResponse response) throws ServletException, IOException {
        response.setContentType("text/html; charset=utf-8");
        Date now = new Date();

        PrintWriter writer = response.getWriter();
        writer.println("<html>");
        writer.println("<head><title>현재 시간</title></head>");
        writer.println("<body>");
        writer.println("<b>현재 날짜는:</b>");
        writer.println(now.toString());
        writer.println("<hr></body>");
        writer.println("</html>");
        writer.close();
    }
}
```



```
protected void doGet(HttpServletRequest req,
                      HttpServletResponse resp)
    throws ServletException,
        java.io.IOException
```

웹브라우저에게 응답을 보낼 때 응답 결과를 html 문서형식으로,
문자 인코딩은 한국어(utf-8)를 사용하겠다.

출력값에 대한 한글 인코딩



예제

- 실행방법

[1] 도스창 아래 경로에서 컴파일

c:\WTomcat 8.0

WwebappsWtestWWEB-INFWsrc>javac -d ../classes NowServlet.java

[2] 웹브라우저 주소창에서 NowSevlet을 요청

http://localhost:9090/test/NowServlet

- webapps/test/

=> <http://ip:port/test/>

web.xml

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://java.sun.com/xml/ns/javaee"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
  http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">
```

```
  <servlet>
    <servlet-name>nowsv</servlet-name>
    <servlet-class>com.test.NowServlet</servlet-class>
  </servlet>
```

```
  <servlet-mapping>
    <servlet-name>nowsv</servlet-name>
    <url-pattern>/now</url-pattern>
  </servlet-mapping>
```

```
</web-app>
```

```
<servlet>
  <servlet-name>invoker</servlet-name>
  <servlet-class>
    org.apache.catalina.servlets.InvokerServlet
  </servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>invoker</servlet-name>
  <url-pattern>/servlet/*</url-pattern>
</servlet-mapping>
```

NowServlet 클래스를 'nowsv' 라는 이름으로 등록

URL이 /now인 경우 nowsv 서블릿이 처리하도록 매핑

http://localhost:9090/test/**NowServlet**
http://localhost:9090/test/**now**



한글처리

■ 1. 입력값에 대한 처리

```
import java.net.*;  
URLDecoder.decode(request.getParameter("name"));
```

또는
`request.setCharacterEncoding("utf-8");`

■ 2. 출력값에 대한 처리

```
response.setContentType("text/html;charset=utf-8");
```



get방식으로 데이터 전송하기

[getTest.html]

```
<!DOCTYPE HTML>
<HTML>
  <HEAD>
    <TITLE> getTest.html </TITLE>
  </HEAD>

  <BODY>
    <!--GET방식으로 데이터 전송
    사용자가 입력한 데이터는 action에 지정된 GetServlet이라는
    서블릿 페이지로 전달됨-->절대 참조 방식
    <form action="/mystudy/PostServlet" method="POST">
    <!-- <form action="/mystudy/GetServlet" method="GET" > -->
      아이디: <INPUT TYPE="text" NAME="id"><br>
      비밀번호:<INPUT TYPE="password" NAME="pwd"><br>
    <INPUT TYPE="submit" value="로그인"><INPUT TYPE="reset" value="취소">
    </form>
  </BODY>
</HTML>
```



GetServlet.java

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

@WebServlet("/GetServlet")
public class GetServlet extends HttpServlet
{
    public void doGet(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException
    {
        res.setContentType("text/html; charset=utf-8");
        PrintWriter pw = res.getWriter();
        //req를 이용하여 사용자가 입력한 값을 받아온다.
        String sid = req.getParameter("id");
        String spwd = req.getParameter("pwd");

        pw.println("<html><body>");
        pw.println(sid + "님 환영합니다.");
        pw.println("</body></html>");

        pw.close();
    }
}
```



GetServlet

- GET 방식으로 요청시
 - `http://localhost:9090/mystudy/GetServlet?id=hong&pwd=1234`
- 쿼리 스트링(Query String)
 - `?id=hong&pwd=1234`
- GET방식
 - 사용자가 입력한 데이터가 요청라인에 그대로 노출됨
 - 데이터 전송방식 - URL 뒤에 쿼리스트링을 붙여 보냄
 - Http Header에 정보를 실어 보냄
- 한글처리
 - get방식으로 보낸 id가 한글인 경우 처리하기 위해서
 - `server.xml`에 `URIEncoding="utf-8"` 추가

```
<Connector port="9090" protocol="HTTP/1.1"  
    connectionTimeout="20000"  
    redirectPort="8443"  
    URIEncoding="utf-8"  
>
```

```
c:\ Tomcat 8.0\ conf\ server.xml
```



PostServlet

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import javax.servlet.annotation.WebServlet;
@WebServlet("/PostServlet")
public class PostServlet extends HttpServlet{
    public void doPost(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException
    {
        res.setContentType("text/html; charset=utf-8");
        PrintWriter pw = res.getWriter();

        req.setCharacterEncoding("utf-8");
        //post 방식에서 사용자가 입력한 내용(request)에 대한 한글처리

        String sid = req.getParameter("id");
        String spwd = req.getParameter("pwd");

        pw.println("<html><body>");
        pw.println(sid + "님 환영합니다.");
        pw.println("</body></html>");

        pw.close();
    }
}
```

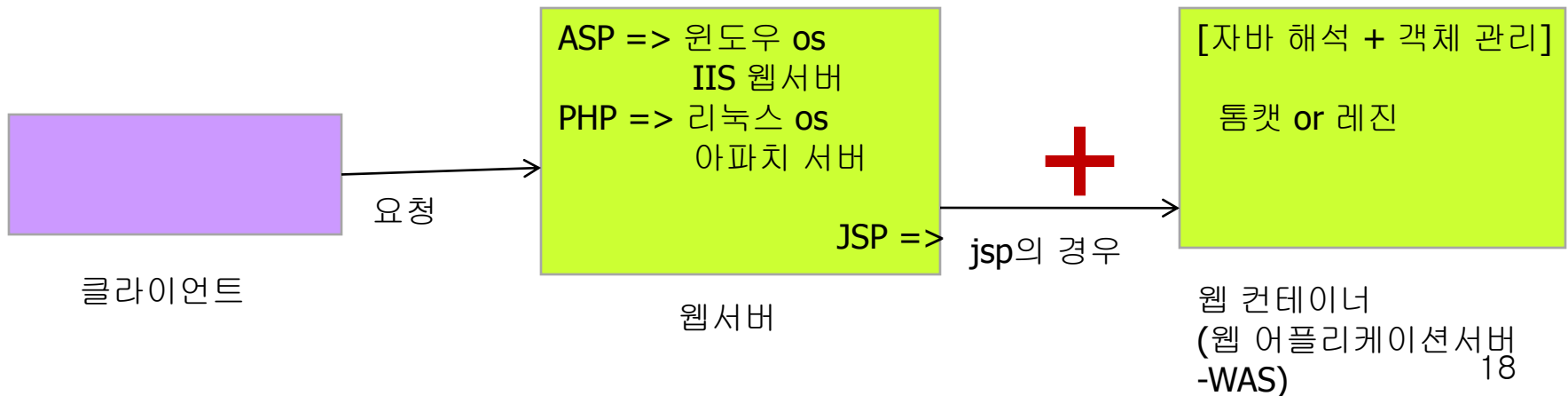



PostServlet

- POST 방식으로 요청시
 - `http://localhost:9090/mystudy/PostServlet`
- POST 방식
 - Http의 Body 부분에 데이터가 포함되어 전송됨
 - 길이 제한이 없다
 - 파일 업로드시 사용
 - 보안에 좋다
- GET 방식
 - 요청라인에 데이터가 전송됨
 - 데이터 길이에 제한 (256byte)

웹 컨테이너

- JSP는 사실상 서블릿이라 불리는 클래스로 변환되어 실행되는 자바언어의 일부
 - JSP로 웹서비스를 구축하려면 웹서버가 java 언어를 이해해야 하지만, 클래스의 바이트 코드를 이해하는 소프트웨어는 오직 JVM이므로, 대부분의 웹서버는 자바 언어를 해석할 수 있도록 설계되어 있지 않다.
 - 따라서 자바 기반의 웹서비스를 구축하려면 **서블릿 클래스의 해석 및 관련 객체들을 관리**해주는 특정 프로그램의 도움을 받지 않으면 안되는데 이 역할을 해주는 프로그램을 **컨테이너**라고 하며, 톰캣, 레진등이 있다.





웹 컨테이너

- 웹 컨테이너 : 예) 톰캣

- 서블릿을 관리

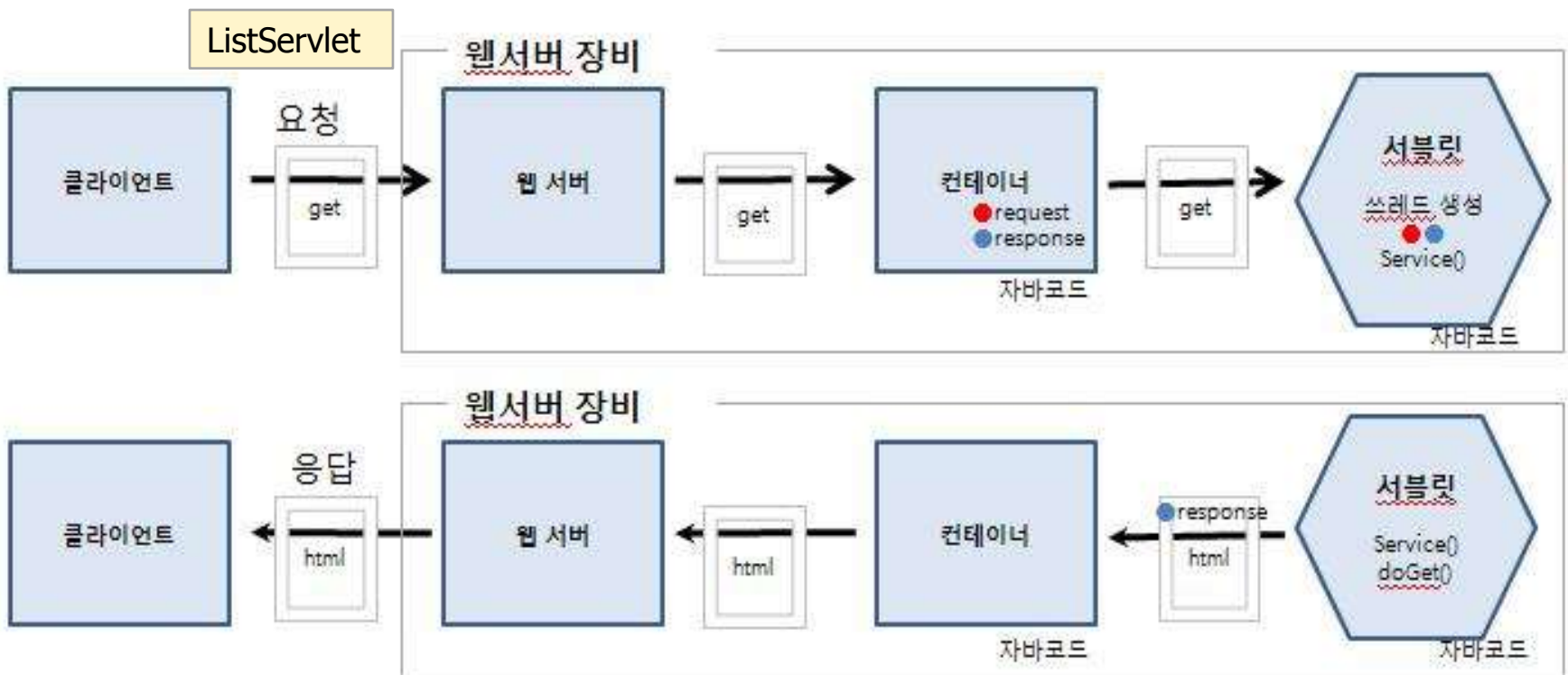
- 서블릿에 대한 요청을 받고 응답을 해주는 중간 역할
 - 클라이언트와 서블릿간의 요청과 답변을 전달해줌
 - 요청을 넘겨받은 컨테이너는 Http Request와 Http response 객체를 만들어 서블릿의 doPost(), doGet() 메서드 중 하나를 호출한다.

- 생명주기 관리

- 서블릿 클래스를 로딩하여 인스턴스화
 - 초기화 메소드를 호출 - init()
 - 요청이 들어오면 적절한 서블릿 메소드를 호출 - service(), doGet(), doPost()

웹 컨테이너

- 클라이언트의 요청에 대한 처리





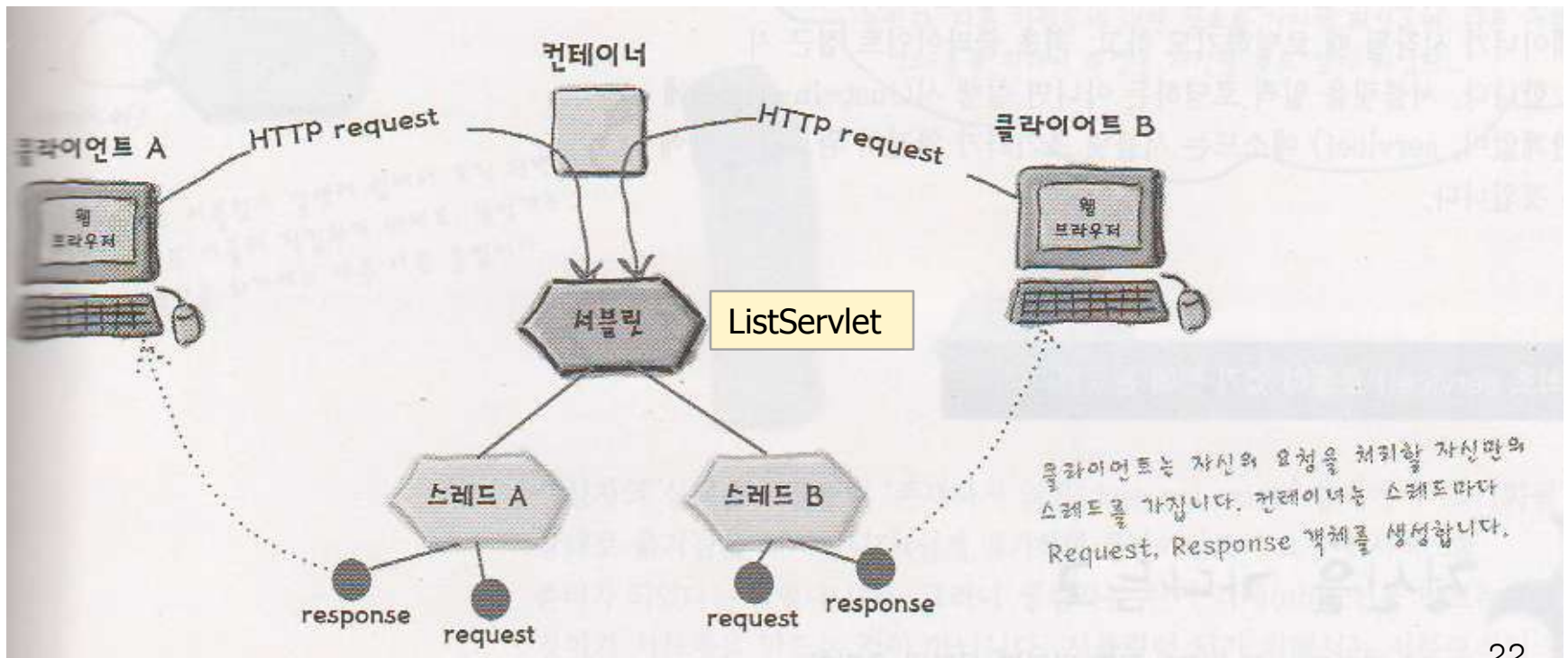
웹 컨테이너

- 1. 사용자가 서블릿에 대한 링크를 클릭
- 2. 웹 컨테이너는 들어온 요청이 서블릿이라는 것을 간파하고 `HttpServletResponse`와 `HttpServletRequest`객체를 생성
- 3. 사용자가 날린 URL을 분석하여 어떤 서블릿에 대한 요청인지 알아냄.(여기서 DD : web.xml를 참조)
 - 그 다음 해당 서블릿 스레드를 생성하여 Request/Response객체를 인자로 넘김
- 4. 웹 컨테이너는 서블릿 `service()` 메소드를 호출. 요청에 지정한 방식(method)에 따라 `doGet()`을 호출할지, `doPost()`를 호출할지 결정
- 5. `doGet()/doPost()`메소드는 동적인 페이지를 생성한 다음, 이를 `Response`객체에 실어 보냄
 - 보내고 난 후에도 웹 컨테이너는 `Response`객체에 대한 렌퍼런스값(참조값)을 가지고 있다.
- 6. 스레드 작업이 끝나면, 웹 컨테이너는 `Response`객체를 HTTP Response로 전환하여 클라이언트로 내려 보냄
 - 마지막으로 `Request`와 `Response`객체를 소멸 시킴

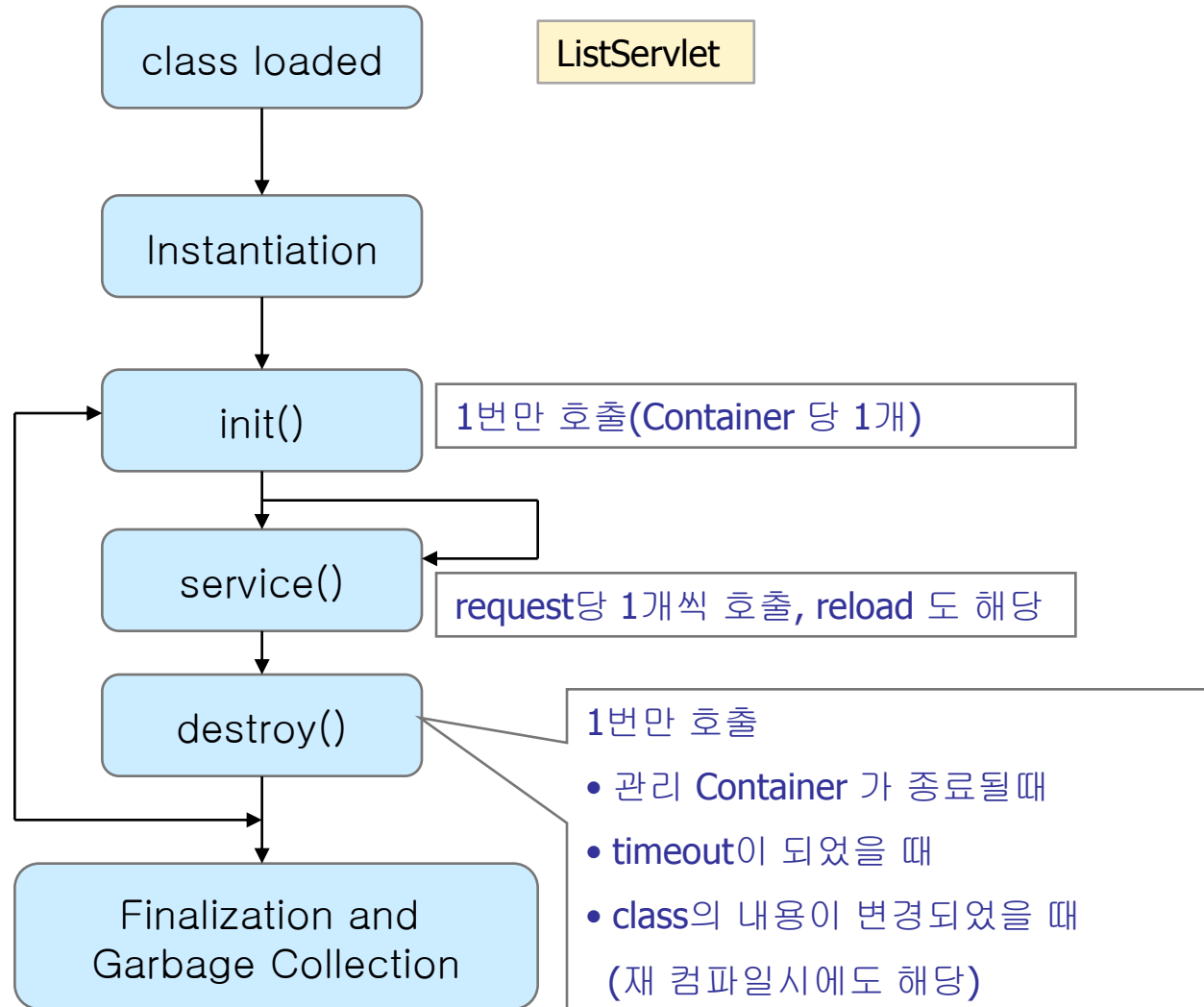
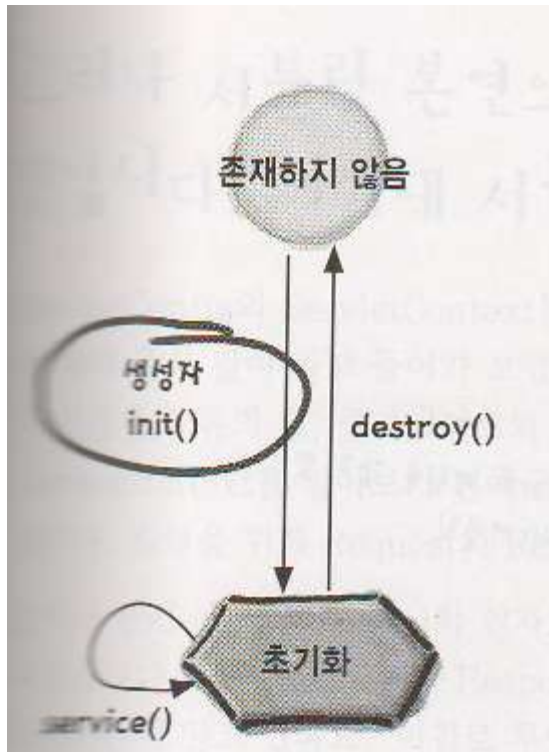
서블릿

■ 서블릿

- 자바의 클래스 중 오직 서버에서만 실행될 수 있는 클래스를 의미함
- 클라이언트의 요청을 받아 처리하여 응답하는 게 주요 업무
- HttpServlet을 상속받아서 만든다.
- 최초의 브라우저 접속에 의해 인스턴스가 메모리에 올라감.



서블릿 라이프 사이클





서블릿 라이프 사이클

- 서블릿이 처음에 로드되면 `init()`메서드가 호출됨
 - `init()`메서드에서는 서블릿이 서비스하기 위해 필요한 초기화 작업을 수행하고, `init()` 메서드가 `service()`메서드를 호출함
 - 초기화된 서블릿은 클라이언트의 요청이 있을 때마다 쓰레드가 생성 되어서 병행적으로 `service()` 메서드를 실행함
 - 서블릿은 더 이상 서비스를 하지 않을 경우에는 서블릿 엔진에 의해 메모리에서 unload됨
 - unload 되기 전에 `destroy()`메서드가 실행됨

웹 컨테이너

서블릿 클래스

서블릿 객체



클래스 로딩

```
101101
101101
101101
10101000010
1010 10 0
01010 1
1010101
10101010
1001010101
```

AServlet.class

서블릿 인스턴스화(생성자 실행)

init()

service()

destroy()

여기서 일생의 대부분을 보냅니다.

초기화

초기화

디폴트 생성자(인자가 없는) 실행(생성자를 코딩하지 마세요. 컴파일러가 자동으로 만들어주는 것을 사용하면 됩니다)

서블릿 일생 중 한번만 호출됨. init() 메소드는 service() 메소드 전에 실행되어야 합니다.

클라이언트 요청을 핸들링 함

doGet(), doPost() 등.
요청이 들어올 때마다 새로운 스레드에서 실행됨.

서블릿이 죽기 전에 가지고 있는 자원들을 깨끗이 정리할 기회를 줍니다. init() 메소드처럼 오직 한번만 호출됩니다.

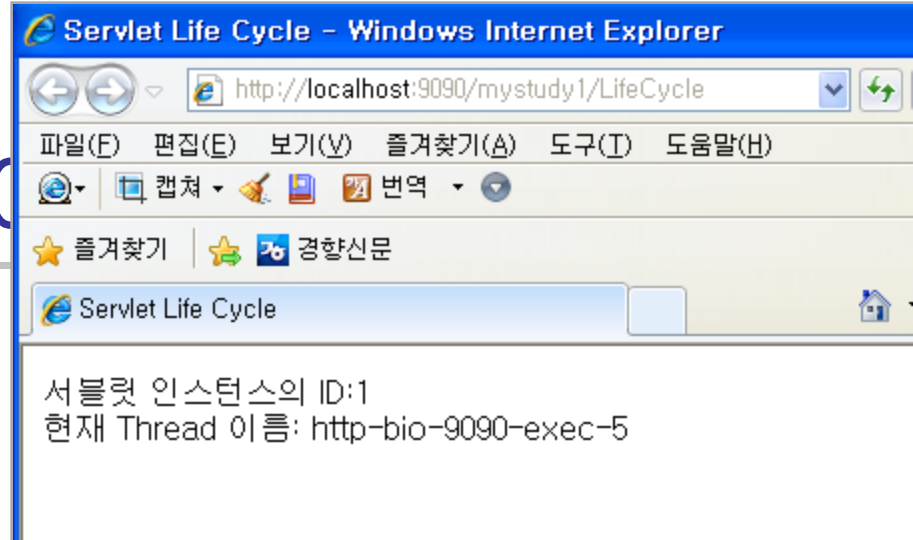
서블릿 라이프 사이클

```
@WebServlet("/LifeCycle")
public class LifeCycle extends HttpServlet {
    private static final long serialVersionUID = 1L;
    public static int num=0;
    public int id;

    public LifeCycle() {
        super();
        System.out.println("생성자");
    }

    public void init(ServletConfig config) throws ServletException {
        super.init(config);
        num++;
        id=num;
        System.out.println("init() : " + id);
    }

    public void destroy() {
        System.out.println("destroy()!!");
    }
}
```



```
생성자
init() : 1
doGet ()
doGet ()
doGet ()
```

ServletConfig-서블릿 배포시 설정된 정보를
서블릿으로 넘겨주기 위하여 사용
서블릿 당 **ServletConfig** 개체 하나



서블릿 라이프 사이클

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
    ServletException, IOException {
    System.out.println("doGet()");
    response.setContentType("text/html;charset=utf-8");
    PrintWriter out=response.getWriter();
    out.println("<HTML><HEAD><title>Servlet Life Cycle</title></head>");
    out.println("<body>");
    out.println("서블릿 인스턴스의 ID:"+id+ "<br>");
    Thread th = Thread.currentThread();
    out.println("현재 Thread 이름: "+th.getName()+"<br>");
    out.println("</body></html>");
}
```



init()

web.xml

```
<servlet>
  <servlet-name>myDispatcherServlet</servlet-name>
  <servlet-class>com.controller.MyDispatcherServlet</servlet-class>
  <init-param>
    <param-name>configFile</param-name>
    <param-value>/config/Command.properties</param-value>
  </init-param>
</servlet>
```

com.controller.MyDispatcherServlet

```
//해당 서블릿이 요청될때 최초로 한번만 호출되는 메서드
public void init(ServletConfig config){
    //매핑 파일을 읽어서 Properties 컬렉션에 담아 놓는다
    //web.xml에서 <init-param>의 값 읽기-CommandPro.properties 파일
    configFile
    = config.getInitParameter("configFile"); //=>/config/CommandPro.properties
    ....
}
```



서블릿 예제

- 서블릿을 이용하여 상품 테이블에 insert 하기

```
drop table pd;  
create table pd  
(  
    no number primary key,  
    pdName varchar2(50) not null,  
    price number null,  
    regdate date default sysdate  
);  
  
create sequence pd_seq  
increment by 1  
start with 1  
nocache;  
  
insert into pd(no, pdname, price)  
values (pd_seq.nextval, '마우스', 15000) ;
```



registerPd.html

```
<HTML>
<HEAD>
  <TITLE> registerPd.html </TITLE>
</HEAD>

<BODY>
<form name="frm" method="POST" action="/mystudy/InsertServlet">
  상품명: <INPUT TYPE="text" NAME="pdName"><br>
  가격:<INPUT TYPE="text" NAME="price"><br>
<INPUT TYPE="submit" value="등록">
<INPUT TYPE="reset" value="취소">
</form>
</BODY>
</HTML>
```



서블릿 예제-상품등록

```
package com.study;
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
@WebServlet("/InsertServlet")
public class InsertServlet extends HttpServlet{
    String url ="jdbc:oracle:thin:@121.173.144.34:1521:ORCL";
    String uid = "javauser1"; String upwd="java";
    Connection con;
    PreparedStatement pstmt;

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



상품등록

```
public void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException{
    response.setContentType("text/html; charset=utf-8");
    PrintWriter out = response.getWriter();
    request.setCharacterEncoding("utf-8");
    String pdname = request.getParameter("pdName"); //상품명
    String price = request.getParameter("price"); //가격
    out.print("상품명 : " + pdname+"<br>");

    try {    //오라클 연동 시작!!
        Class.forName("oracle.jdbc.driver.OracleDriver");
        out.print("driver load 성공! <br>");

        con = DriverManager.getConnection(url, uid, upwd);
        if(con!=null){
            out.println("connection 성공!! <br>");
            String sql = "insert into pd(no, pdname, price)"
                +"values(pd_seq.nextval,?,?)";
            pstmt=con.prepareStatement(sql);
            pstmt.setString(1, pdname);
            pstmt.setString(2, price);
```




상품등록

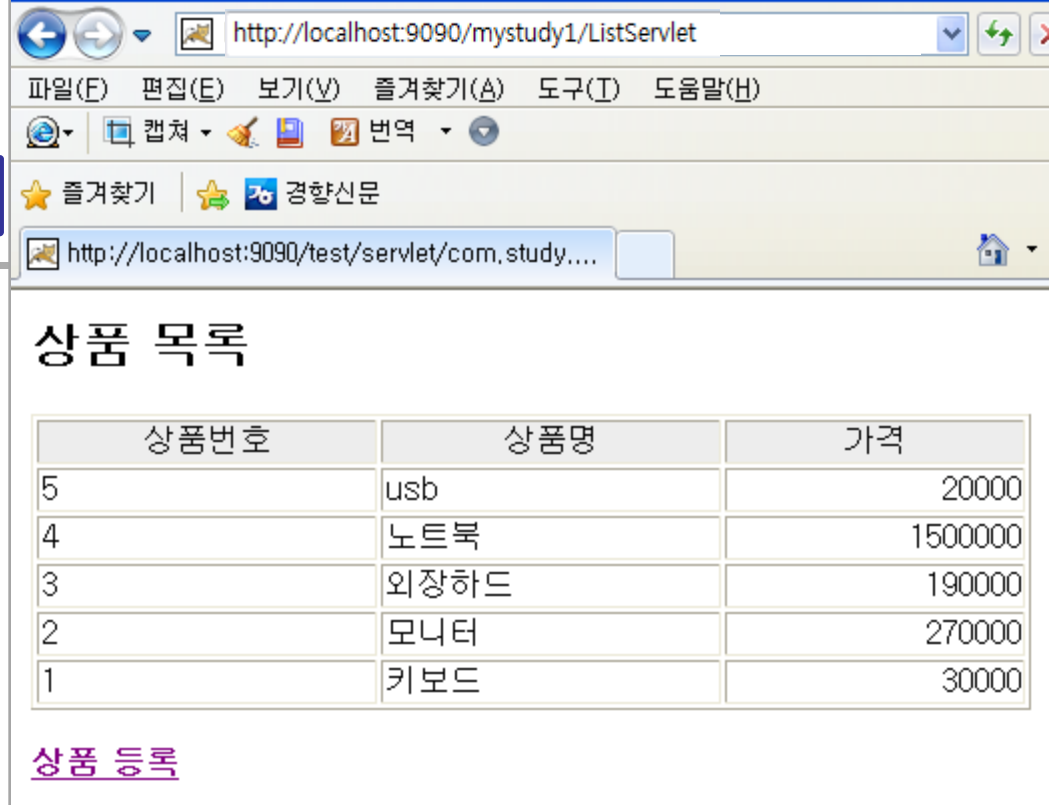
```
int result = pstmt.executeUpdate();
if(result>0){
    out.print("insert 성공!! <br>");
    response.sendRedirect("ListServlet");
}
else out.print("insert 실패!! <br>");
}else{
    out.println("connection 실패!! <br>");
}
} catch (SQLException e) { e.printStackTrace();
} catch (ClassNotFoundException e) {
    out.print("driver load 실패! <br>");
    e.printStackTrace();
}finally{
    try {
        if(pstmt!=null) pstmt.close();
        if(con!=null) con.close();
    } catch (SQLException e) { e.printStackTrace(); }
}
}
} //
```

상품 목록 조회

```
package com.study;
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Timestamp;
import java.text.DecimalFormat;
import java.text.SimpleDateFormat;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
```

```
@WebServlet("/ListServlet")
```

```
public class ListServlet extends HttpServlet {
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
        ServletException, IOException {
        //응답문서의 컨텐츠 타입 지정
        response.setContentType("text/html;charset=utf-8");
```



상품번호	상품명	가격
5	usb	20000
4	노트북	1500000
3	외장하드	190000
2	모니터	270000
1	키보드	30000

[상품 등록](#)

```
PrintWriter out = response.getWriter();
DecimalFormat df = new DecimalFormat("#,###");
SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
```

```
//3. db작업 - select
```

```
Connection con=null;
```

```
PreparedStatement ps=null;
```

```
ResultSet rs=null;
```

```
try{
```

```
    //[1] 드라이버 로딩
```

```
    Class.forName("oracle.jdbc.driver.OracleDriver");
```

```
    System.out.println("드라이버 로딩 성공");
```

```
    //[2] 연결객체
```

```
    String url="jdbc:oracle:thin:@yang-hp:1521:orcl";
```

```
    String uid="javauser1", upwd="java";
```

```
    con=DriverManager.getConnection(url, uid, upwd);
```

```
    System.out.println("db연결 성공");
```

```
    //[3] sql 문장 처리
```

```
    String sql="select * from pd order by no desc";
```

```
    ps=con.prepareStatement(sql);
```

```
    //[4] 실행
```

```
    rs = ps.executeQuery();
```

```
    out.println("<h2>상품 목록</h2>");
```

```
    out.println("<table width='500' border='1'>");
```

```

        out.println("<tr><th>번호</th>");
        out.println("<th>상품명</th>");
        out.println("<th>가격</th>");
        out.println("<th>등록일</th></tr>");

        //반복 시작
        while(rs.next()){
            int no=rs.getInt("no");
            String pdName=rs.getString("pdName");
            int price = rs.getInt("price");
            Timestamp regdate = rs.getTimestamp("regdate");

            out.println("<tr><td>" + no+"</td>");
            out.println("<td><a href='DetailServlet?no="+no+"'>"
                        + pdName + "</a></td>");
            out.println("<td align='right'>" + df.format(price)+"원</td>");
            out.println("<td align='center'>" + sdf.format(regdate)+"</td></tr>");
        }//while
        //반복 끝
        out.println("</table>");
        System.out.println("상품 전체 목록 조회 성공");
    }catch(ClassNotFoundException e){
        System.out.println("class not found!");
        e.printStackTrace();
    }catch(SQLException e){
        System.out.println("sql 에러");
        e.printStackTrace();
    }

```

```

}finally{
    try {
        if(rs!=null) rs.close();
        if(ps!=null) ps.close();
        if(con!=null) con.close();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

```

//4. 결과처리

```

out.println("<br><a href='/mystudy/svTest/registerPd.html'>상품 등록</a>");

```

```

}

```

```

}

```



상품 상세 보기

```
@WebServlet("/DetailServlet")
public class DetailServlet extends HttpServlet {
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
        ServletException, IOException {
        response.setContentType("text/html; charset=utf-8");

        PrintWriter out = response.getWriter();

        String no = request.getParameter("no");
        Connection con=null;
        PreparedStatement pstmt=null;
        ResultSet rs=null;
        String pdName = "";
        int price=0;
        Timestamp regdate=null;

        try {
            Class.forName("oracle.jdbc.driver.OracleDriver");
            System.out.println("driver load 성공!");
```

```

//접속
String url ="jdbc:oracle:thin:@yang2:1521:ORCL";
String uid = "javauser1";
String upwd ="java";
con = DriverManager.getConnection(url, uid, upwd);
System.out.println("connection 성공!!<br>");
String sql = "select * from pd where no=?";
pstmt=con.prepareStatement(sql);
pstmt.setInt(1, Integer.parseInt(no));
rs = pstmt.executeQuery();
if(rs.next()){
                pdName = rs.getString("pdname");
                price=rs.getInt("price");
                regdate=rs.getTimestamp("regdate");
            }
        }
    } catch (SQLException e) {
        e.printStackTrace();
    } catch (ClassNotFoundException e) {
        System.out.println("driver load 실패!<br>");
        e.printStackTrace();
    }
}finally{
    try {
        if(rs!=null) rs.close(); if(pstmt!=null) pstmt.close();
        if(con!=null) con.close();
    } catch (SQLException e) {e.printStackTrace(); }
}

```

```

out.println("<h2>상품 상세 보기</h2>");
out.println("번호 : "+ no+"<br>");
out.println("상품명 : " + pdName+ "<br>");
out.println("가격 : "+ df.format(price)+"원<br>");
out.println("등록일 : "+ regdate+"<br><br>");

out.println("<a href='ListPdServlet'>목록</a> | ");
out.println("<a href='EditServlet?no="+no+"'>수정</a> | ");
//out.println("<a href='DeleteServlet?no="+no+"'>삭제</a> | ");

out.println("<a href='#' onclick='del(\""+no+"')'>삭제</a>");

out.println("<script type='text/javascript'>");
out.println("function del(no){");
out.println("if(confirm('삭제 하시겠습니까?')){");
out.println("location.href='DeleteServlet?no='"+no+"';");
out.println("}");
out.println("}");
out.println("</script>");

out.close();          }

} //class

```