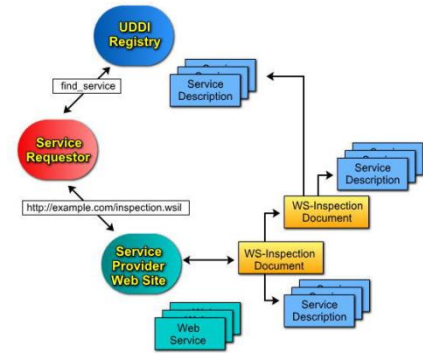


# Web Application & Service

Servlet 객체들



컴퓨터 과학과  
김희열

# ServletConfig 객체

- Servlet이 초기화될 때 관련 정보를 저장해서 제공되는 객체
- Container에 의해 생성되어 Servlet에게 전달됨
- web.xml에서 <init-param>을 읽어서 ServletConfig에 저장

```
<servlet>
  <description></description>
  <display-name>EmailServlet</display-name>
  <servlet-name>EmailServlet</servlet-name>
  <servlet-class>myapp.EmailServlet</servlet-class>
  <init-param>
    <param-name>adminEmail</param-name>
    <param-value>admin@myservice.com</param-value>
  </init-param>
  <init-param>
    <param-name>myEmail</param-name>
    <param-value>heeyoul.kim@kgu.ac.kr</param-value>
  </init-param>
</servlet>
```

# ServletConfig 객체

- Annotation-based config

**Create Servlet**  
Enter servlet deployment descriptor specific information.

Name:

Description:

Initialization parameters:

Name	Value	Description
adminEmail	admin@myservice.com	
myEmail	heeyoul.kim@kgu.ac.kr	

URL mappings:

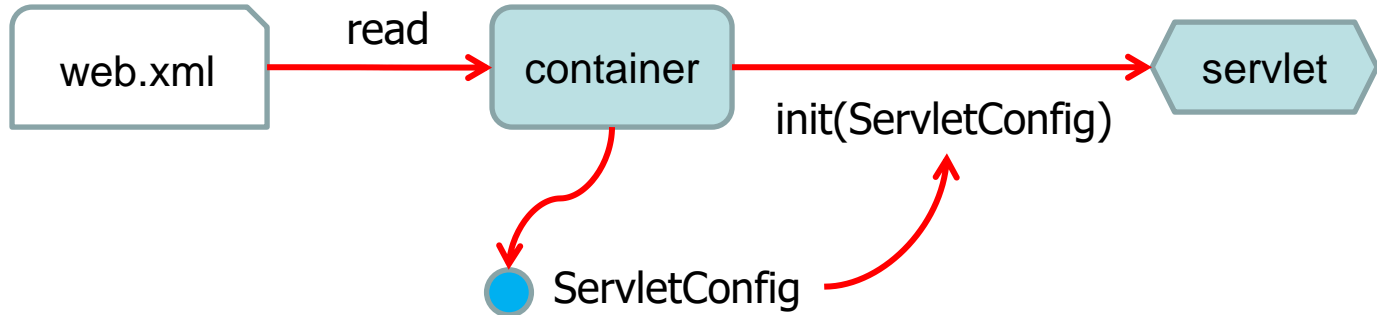
/EmailServlet
---------------

< Back   Next >   Finish   Cancel

```
@WebServlet(  
    urlPatterns = { "/EmailServlet" },  
    initParams = {  
        @WebInitParam(name = "adminEmail", value = "admin@myservice.com"),  
        @WebInitParam(name = "myEmail", value = "heeyoul.kim@kgu.ac.kr")  
    })  
public class EmailServlet extends HttpServlet {
```

# ServletConfig 객체

- Servlet이 초기화될 때



- Methods
  - getInitParameter(name)**
    - 파라미터 중 name에 연관된 value를 리턴
  - getInitParameterNames()**
    - 파라미터의 name들을 묶어서 Enumeration으로 리턴
  - getServletContext()**
    - ServletContext 객체 리턴

# ServletConfig 객체

- EmailServlet.java

```
package myapp;

import java.io.IOException;
import java.io.PrintWriter;
import java.util.Enumeration;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

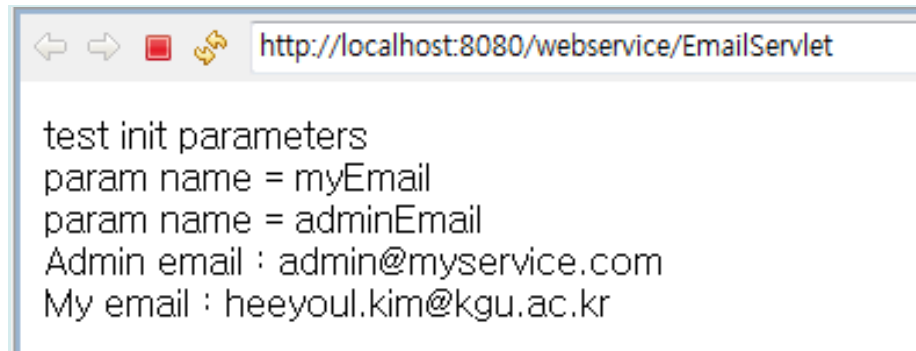
public class EmailServlet extends HttpServlet {
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("test init parameters<br>");

        ServletConfig servletConfig = getServletConfig();
        Enumeration<?> e = servletConfig.getInitParameterNames();
        while (e.hasMoreElements()) {
            out.println("param name = " + e.nextElement() + "<br>");
        }
        out.println("Admin email : " + servletConfig.getInitParameter("adminEmail"));
        out.println("<br>");
        out.println("My email      : " + servletConfig.getInitParameter("myEmail"));
    }
}
```

# ServletConfig 객체

---

- Run & test

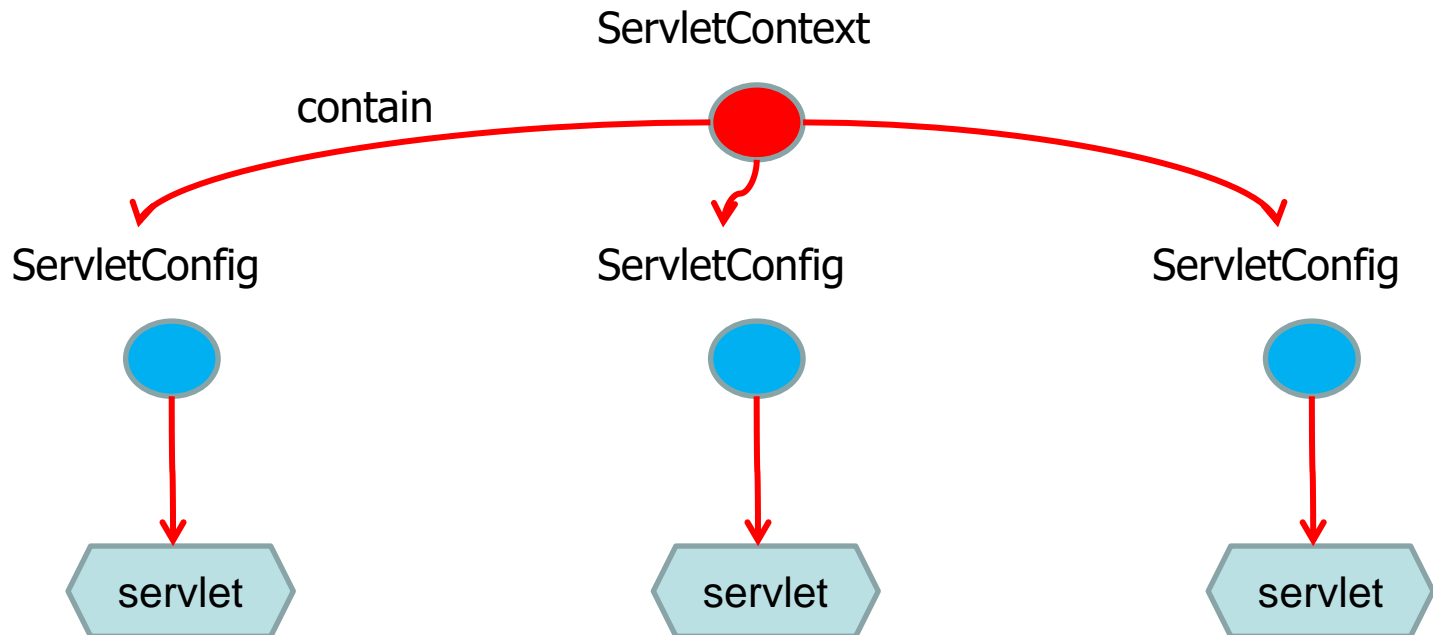


A screenshot of a web browser window. The address bar shows the URL `http://localhost:8080/webservice/EmailServlet`. The main content area displays the following text:

```
test init parameters  
param name = myEmail  
param name = adminEmail  
Admin email : admin@myservice.com  
My email : heeyoul.kim@kgu.ac.kr
```

# ServletContext 객체

- 하나의 web application마다 하나의 ServletContext 객체가 존재
  - Web application : 하나의 서버 URL namespace ([ex] /webservice)아래에 모여 있는 servlet과 content의 집합
- 서버와 container에 관련된 정보를 servlet에 제공
- 포함된 모든 servlet에게 초기 파라미터를 제공할 때 유용



# ServletContext 객체

---

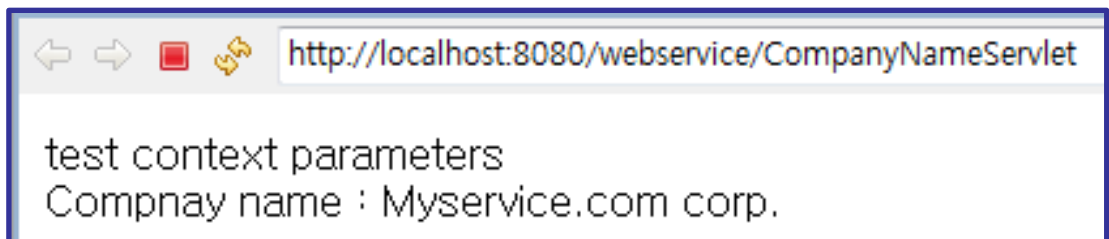
- Methods
  - 서버 정보 관련
    - `getServerInfo()`
    - `getMajorVersion()`
    - ...
  - 서버 자원 정보
    - `getMimeType(filename)`
    - `getResource(path)`
    - ...
  - Logging
    - `log(message)`
    - ...
  - Attribute 관리
    - `getAttribute(name)`
    - `setAttribute(name, value)`
    - ...
  - Parameter
    - `getInitParameter(name)`
    - `getInitParameterNames()`



# ServletContext 객체

- 초기 파라미터 전달 수단으로 활용
  - 모든 servlet이 공유해야 하는 정보 (ex. 회사명, DB server) 전달

```
</servlet-mapping>
<context-param>
  <param-name>companyName</param-name>
  <param-value>Myervice.com corp.</param-value>
</context-param>
<servlet>
```



- Servlet간의 attribute 공유, 컨테이너와의 연결 등을 위해서도 사용

# ServletContext 객체

- CompanyNameServlet.java

```
package myapp;

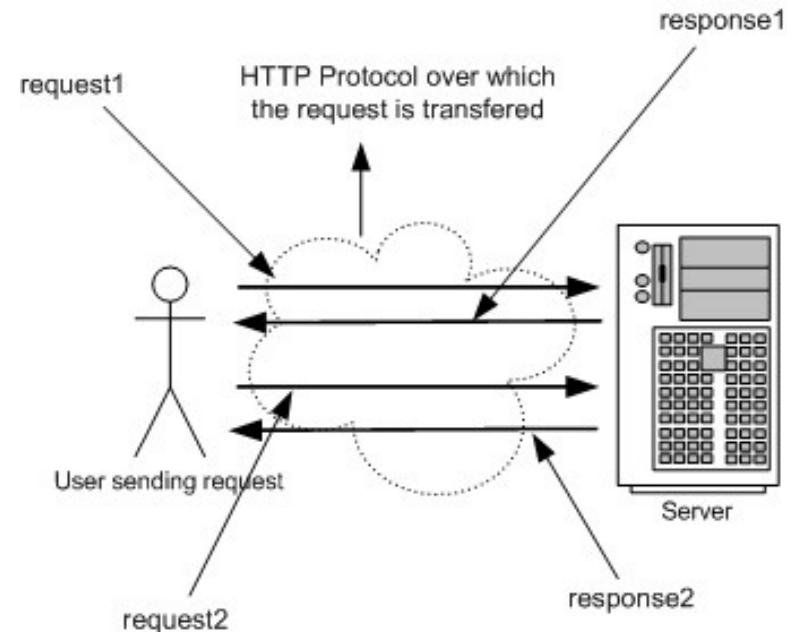
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletConfig;
import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class CompanyNameServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("test context parameters<br>");
        ServletConfig servletConfig = getServletConfig();
        ServletContext servletContext = servletConfig.getServletContext();
        out.println("Compnay name : " + servletContext.getInitParameter("companyName"));
    }
}
```

# HttpSession 객체

- HTTP 프로토콜은 비연결형(stateless) 프로토콜

- 연결 → 요청 → 응답 → 종료
- 연속적인 사용자 정보가 보관되지 않음
  - Ex) 로그인 상태, 장바구니



- 해결방안

- 별도의 수단을 통해 각각의 클라이언트를 구분
- 클라이언트 별로 해당 정보를 유지
- 어떻게 클라이언트를 구분할 것인가?
  - IP 단위로 구분?

# HttpSession 객체

- 쿠키 vs. 세션

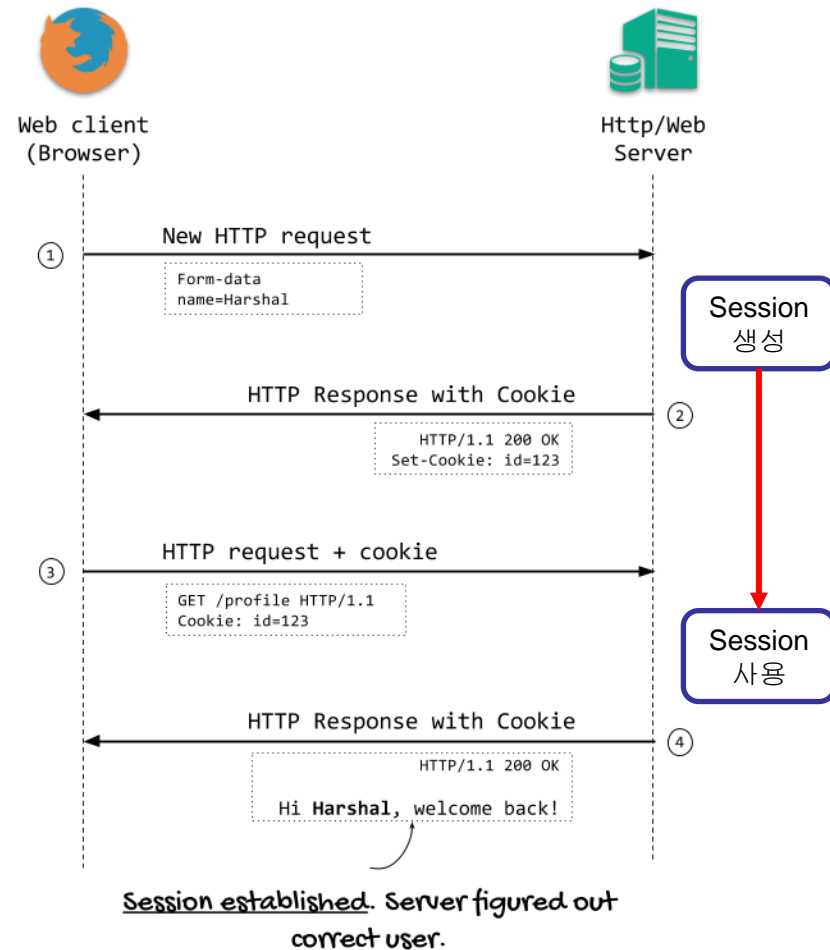
- 쿠키

- 브라우저를 통해 클라이언트에 저장되는 사용자 정보
    - (name, value) 쌍으로 이루어진 정보
    - 초기에 웹 서버에 의해 **HTTP header**에 포함되어 클라이언트에게 전송
    - 이후에 접속마다 클라이언트가 웹 서버에게 재전송
    - 보안적 취약성으로 인해 중요 정보를 저장하지 않아야 함



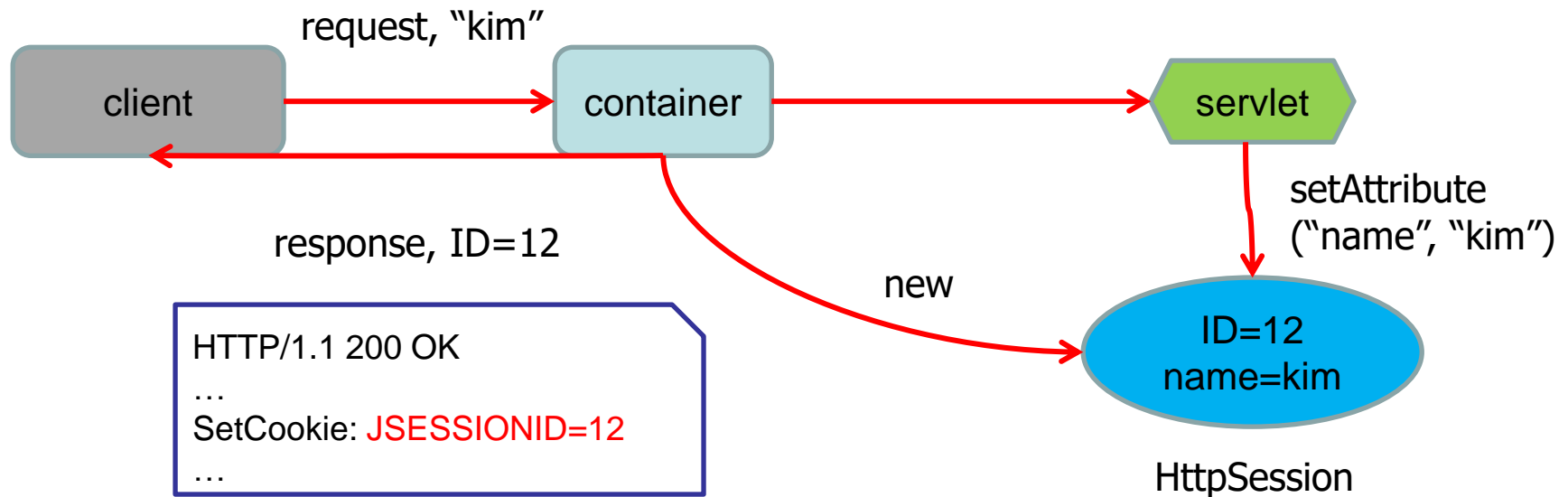
# HttpSession 객체

- 쿠키 vs. 세션
  - 세션
    - 사용자 정보를 서버에 저장
    - 클라이언트의 최초 접속 시 새로운 세션을 생성하고 세션ID 전송
    - 이후 접속마다 클라이언트가 세션 ID 재전송
    - 서버는 세션 ID에 해당하는 세션 정보를 획득
    - 세션 ID 전송 수단으로 쿠키를 사용할 수 있음



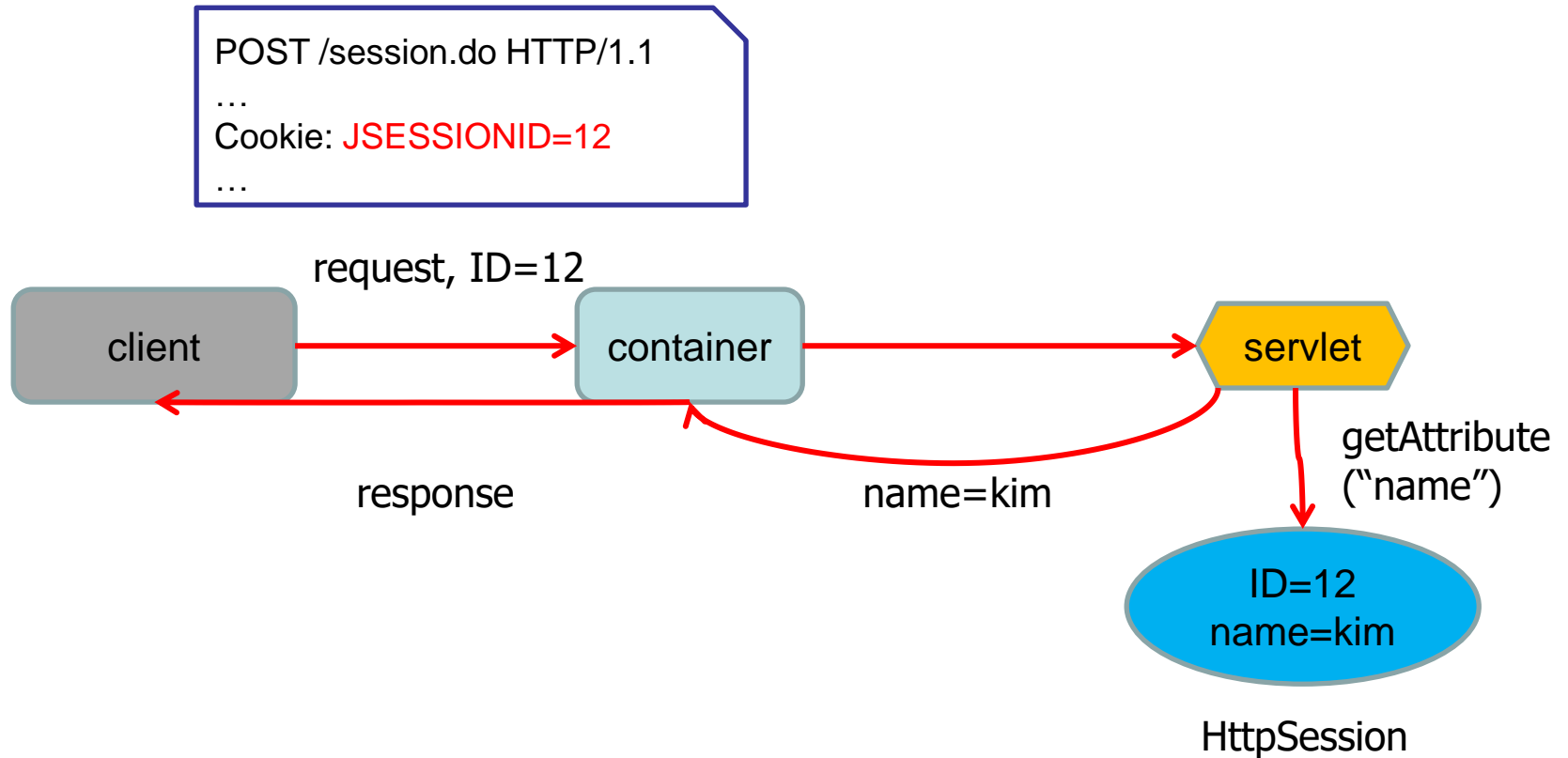
# HttpSession 객체

- 세션 동작 흐름
  - 최초 접속

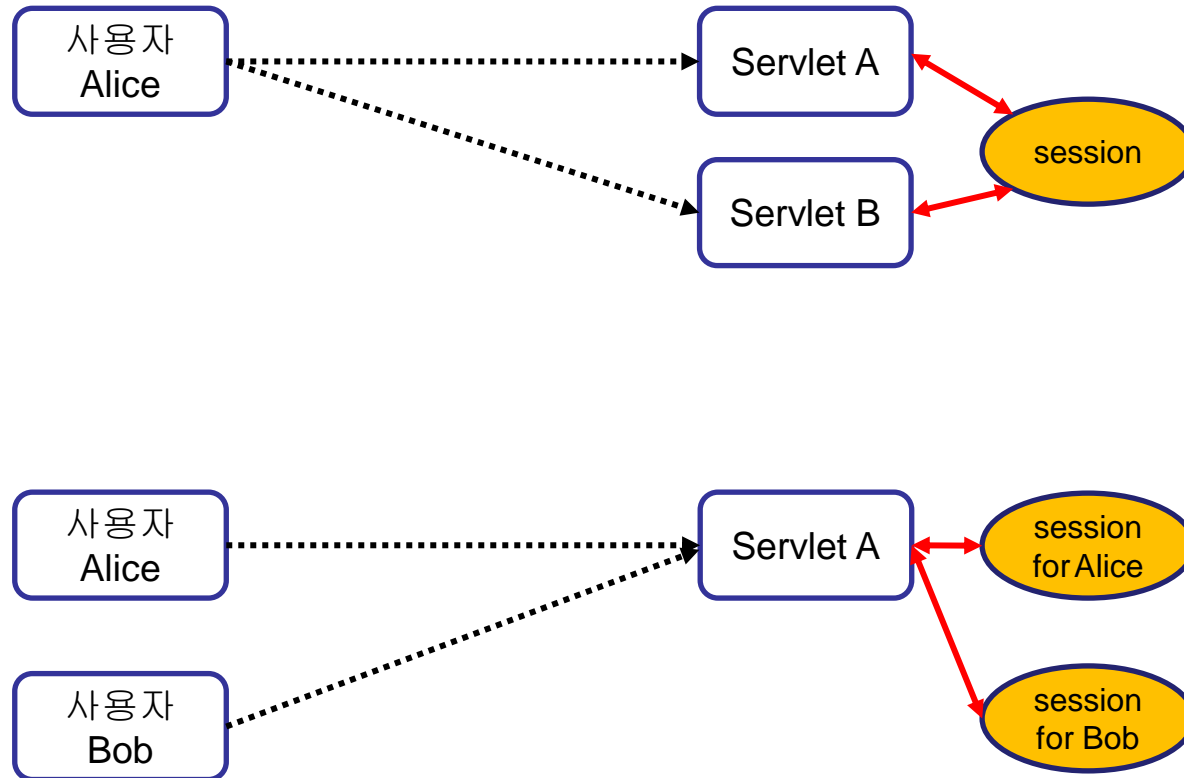


# HttpSession 객체

- 세션 동작 흐름
  - 재접속



# HttpSession 객체





# HttpSession 객체

---

- 세션 관리
  - 대부분의 세션 관련 작업은 container가 처리
  - Servlet은 request로부터 HttpSession 객체를 제공받음

```
HttpSession session = request.getSession();
```

- 생성된 세션이 없었던 경우
  - Container에 의해 처리
  - 새로운 세션 ID 생성
  - 세션 ID를 클라이언트에게 보낼 준비
  - 새로운 HttpSession 객체 생성 후 servlet에 제공
- 기존 세션이 있는 경우
  - Request에 포함된 세션 ID에 해당하는 HttpSession 객체를 찾아서 제공

# HttpSession 객체

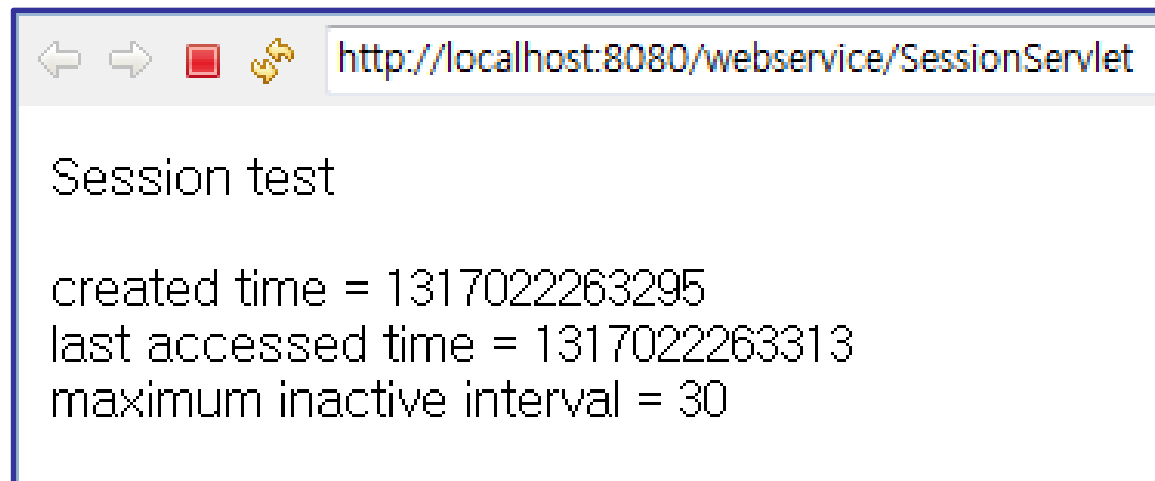
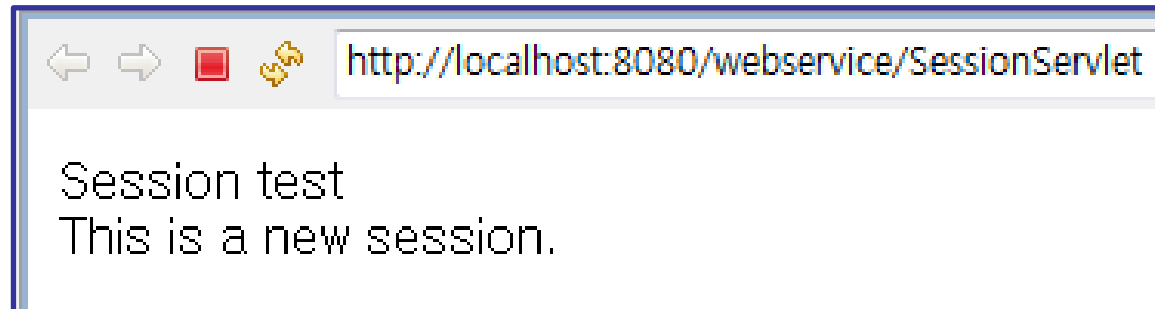
---

- HttpSession Interface

- boolean isNew()
  - 이번 요청으로 새로운 세션이 생성되는지 여부를 리턴
- String getId()
  - 현재의 세션 ID를 리턴
- void invalidate()
  - 현재 세션 종료
- long getCreationTime()
  - 세션이 생성된 시간 제공
- long getLastAccessedTime()
  - 마지막으로 세션에 관련된 클라이언트가 요청을 보낸 시간 제공
- void setMaxInactiveInterval(int)
  - 초단위로 최대 유효기간 설정
- int getMaxInactiveInterval()
  - 초단위로 설정된 최대 유효기간 리턴
- void setAttribute(String, Object)
- Object getAttribute(String)

# HttpSession 객체

- 실습



# HttpSession 객체

- SessionServlet.java

```
@WebServlet("/SessionServlet")
public class SessionServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<HTML><BODY>");
        out.println("Session test<br>");

        HttpSession session = request.getSession();
        if (session.isNew()) {
            out.println("This is a new session.");
            session.setMaxInactiveInterval(30);
        } else {
            out.println("<br>created time = " + session.getCreationTime());
            out.println("<br>last accessed time = " + session.getLastAccessedTime());
            out.println("<br>maximum inactive interval = " + session.getMaxInactiveInterval());
        }
        out.println("</BODY></HTML>");
    }
}
```

# HttpSession 객체

---

- 세션 lifecycle
  - 생성
    - 최초로 request.getSession()이 호출될 때 container가 생성
  - 사용
    - 클라이언트가 세션 ID를 이용해 접속
    - request.getSession()을 통해 사용중인 HttpSession 객체 획득 후 사용
  - 종료
    - invalidate()가 호출되거나, 세션이 타임아웃 되었을 때 container가 소멸

# 속성 (Attribute)

---

- 각 객체의 lifecycle

객체	생성	소멸
ServletContext	웹 application 시작	웹 application 종료
HttpSession	세션을 사용하는 content 최초 접속시	타임아웃 invalidate()호출
HttpServletRequest	해당 servlet 요청시	요청 완료

# 속성 (Attribute)

- 속성

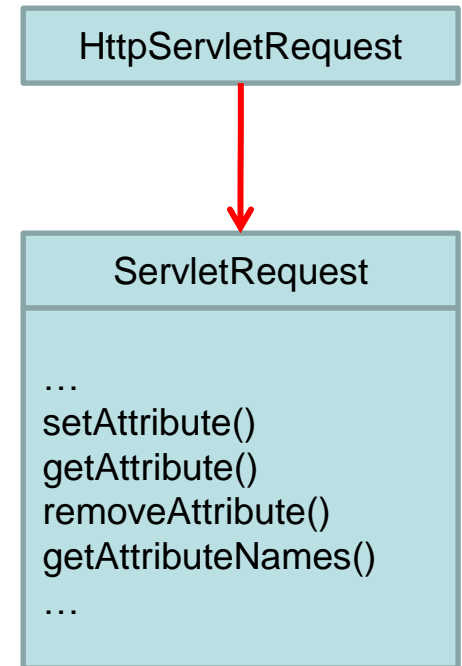
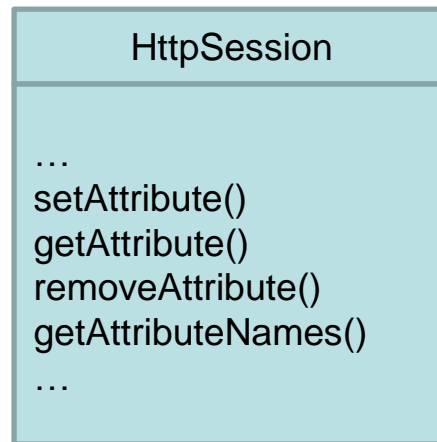
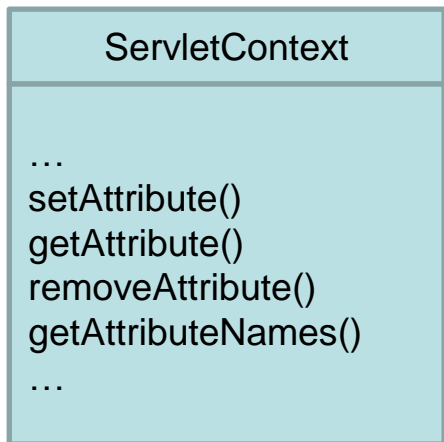
- 특정 정보와 행위를 가지고 있는 객체
- ServletContext, HttpServletRequest, HttpSession 객체에 binding
- Java Object 형태의 객체
  - 모든 java 클래스는 Object 형태로 변환 가능
- 내부적으로 (name, value) 형식으로 저장
- 속성을 binding한 객체의 scope 내에서 정보 공유에 사용
  - Ex) 장바구니

- 속성 scope

객체	속성 사용
ServletContext	동일 application 내에서 모두 사용 가능
HttpSession	동일 session 내에서 사용 가능
HttpServletRequest	동일 request 내에서 사용 가능

# 속성 (Attribute)

- Methods
  - void setAttribute(String name, Object value)
    - 속성 저장
  - Object getAttribute(String name)
    - 속성 추출
  - void removeAttribute(String name)
    - 속성 제거
  - Enumeration getAttributeNames()
    - 속성 name 리스트 획득





# 속성 (Attribute) - ServletContext

- 웹 페이지 방문자수 counter
  - 한 앱 내의 임의의 **servlet**을 방문할 때마다 **count** 증가
  - 서로 다른 사용자가 접근해도 **count**는 계속 증가
  - ServletContext 객체와 속성 이용

## Counter.java

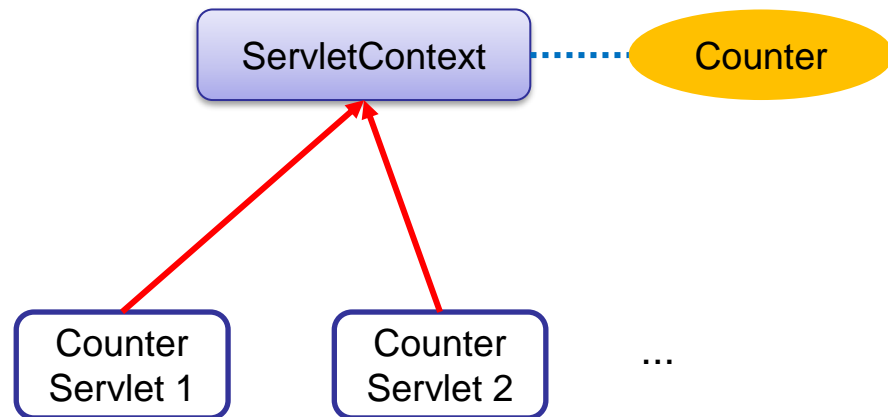
```
package myapp;

public class Counter {
    private int count = 0;

    public void addCount() {
        count++;
    }

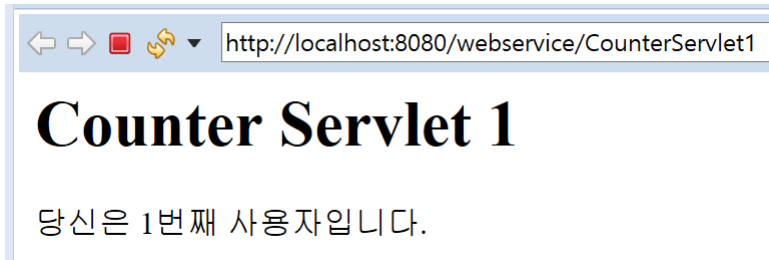
    public void init() {
        count = 0;
    }

    public int getCount() {
        return count;
    }
}
```

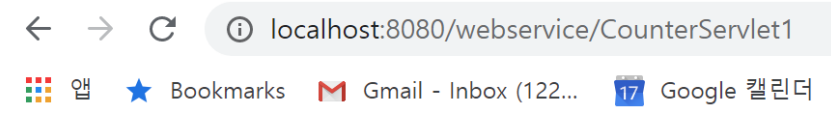


# 속성 (Attribute) - ServletContext

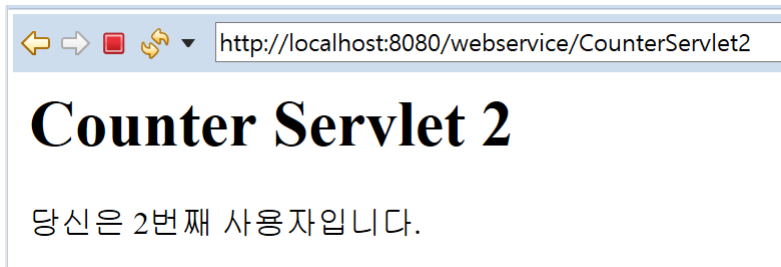
## CounterServlet1 요청



## 다른 사용자가 CounterServlet1 요청



## CounterServlet2 요청



# 속성 (Attribute) - ServletContext

## CounterServlet1.java

```
@WebServlet("/CounterServlet1")
public class CounterServlet1 extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        response.setContentType("text/html; charset=UTF-8");
        PrintWriter out = response.getWriter();
        out.println("<HTML><BODY>");
        out.println("<H1>Counter Servlet 1</H1>");

        ServletContext sc = getServletContext();
        Counter cnt = (Counter)sc.getAttribute("counter");
        if (cnt == null) {
            cnt = new Counter();
        }
        cnt.addCount();
        out.println("<p>당신은 " + cnt.getCount() + "번째 사용자입니다.</p>");
        out.println("</BODY></HTML>");

        sc.setAttribute("counter", cnt);
    }
}
```

# 속성 (Attribute) - ServletContext

## CounterServlet2.java

```
@WebServlet("/CounterServlet2")
public class CounterServlet2 extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        response.setContentType("text/html; charset=UTF-8");
        PrintWriter out = response.getWriter();
        out.println("<HTML><BODY>");
        out.println("<H1>Counter Servlet 2</H1>");

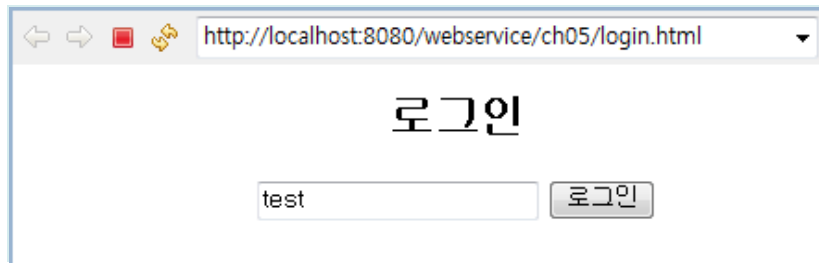
        ServletContext sc = getServletContext();
        Counter cnt = (Counter)sc.getAttribute("counter");
        if (cnt == null) {
            cnt = new Counter();
        }
        cnt.addCount();
        out.println("<p>당신은 " + cnt.getCount() + " 번째 사용자입니다.</p>");
        out.println("</BODY></HTML>");

        sc.setAttribute("counter", cnt);
    }
}
```

# 속성 (Attribute) - HttpSession

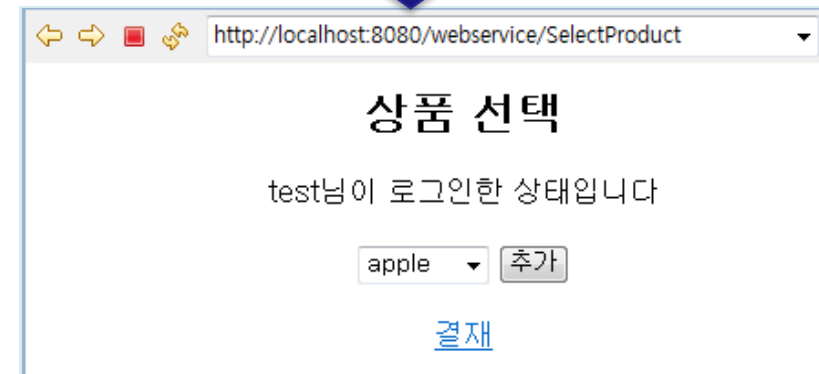
- 장바구니
  - 사용자별로 자신의 이름과 선택한 과일 목록을 저장
  - HttpSession 객체와 속성 이용

/ch05/login.html



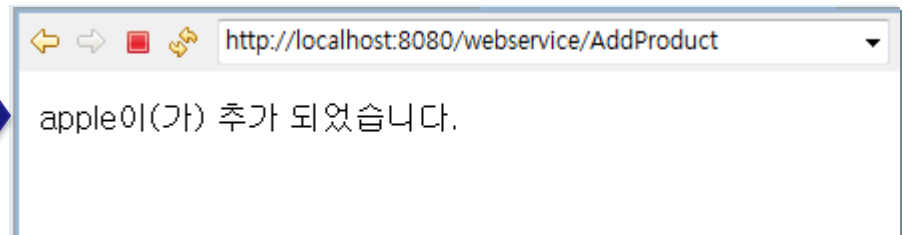
A screenshot of a web browser window showing a login page. The address bar displays 'http://localhost:8080/webservice/ch05/login.html'. The page title is '로그인' (Login). Below the title, there is a text input field containing the text 'test' and a button labeled '로그인' (Login).

SelectProduct



A screenshot of a web browser window showing a product selection page. The address bar displays 'http://localhost:8080/webservice/SelectProduct'. The page title is '상품 선택' (Product Selection). Below the title, it says 'test님이 로그인한 상태입니다' (You are logged in as test). There is a dropdown menu showing 'apple' and a button labeled '추가' (Add). At the bottom, there is a blue link labeled '결제' (Payment).

AddProduct



A screenshot of a web browser window showing a confirmation page. The address bar displays 'http://localhost:8080/webservice/AddProduct'. The page content says 'apple이(가) 추가 되었습니다.' (apple has been added).

# 속성 (Attribute) - HttpSession

SelectProduct



상품 선택

test님이 로그인한 상태입니다

lemon ▼ 추가

[결재](#)

AddProduct



lemon이(가) 추가 되었습니다.

CheckOut

<http://localhost:8080/webservice/CheckOut>

상품 선택

test님이 로그인한 상태입니다

lemon ▼ 추가

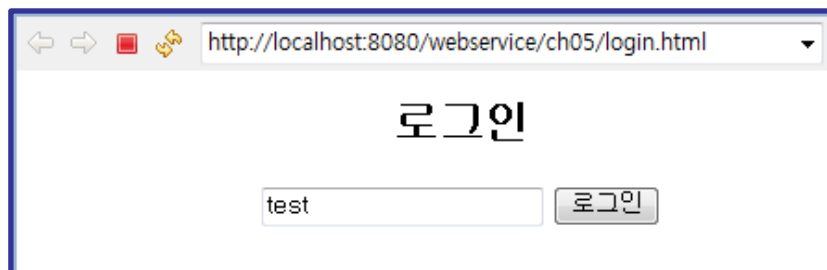
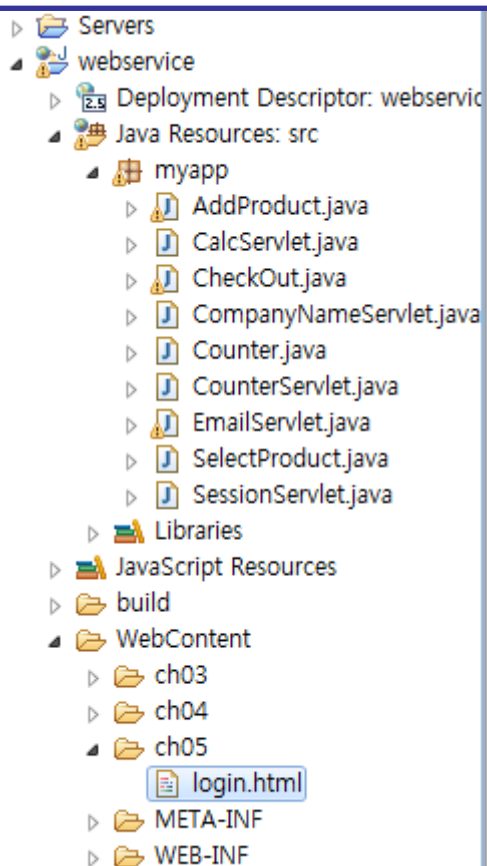
[결재](#)



test님이 선택한 상품 목록

apple  
lemon

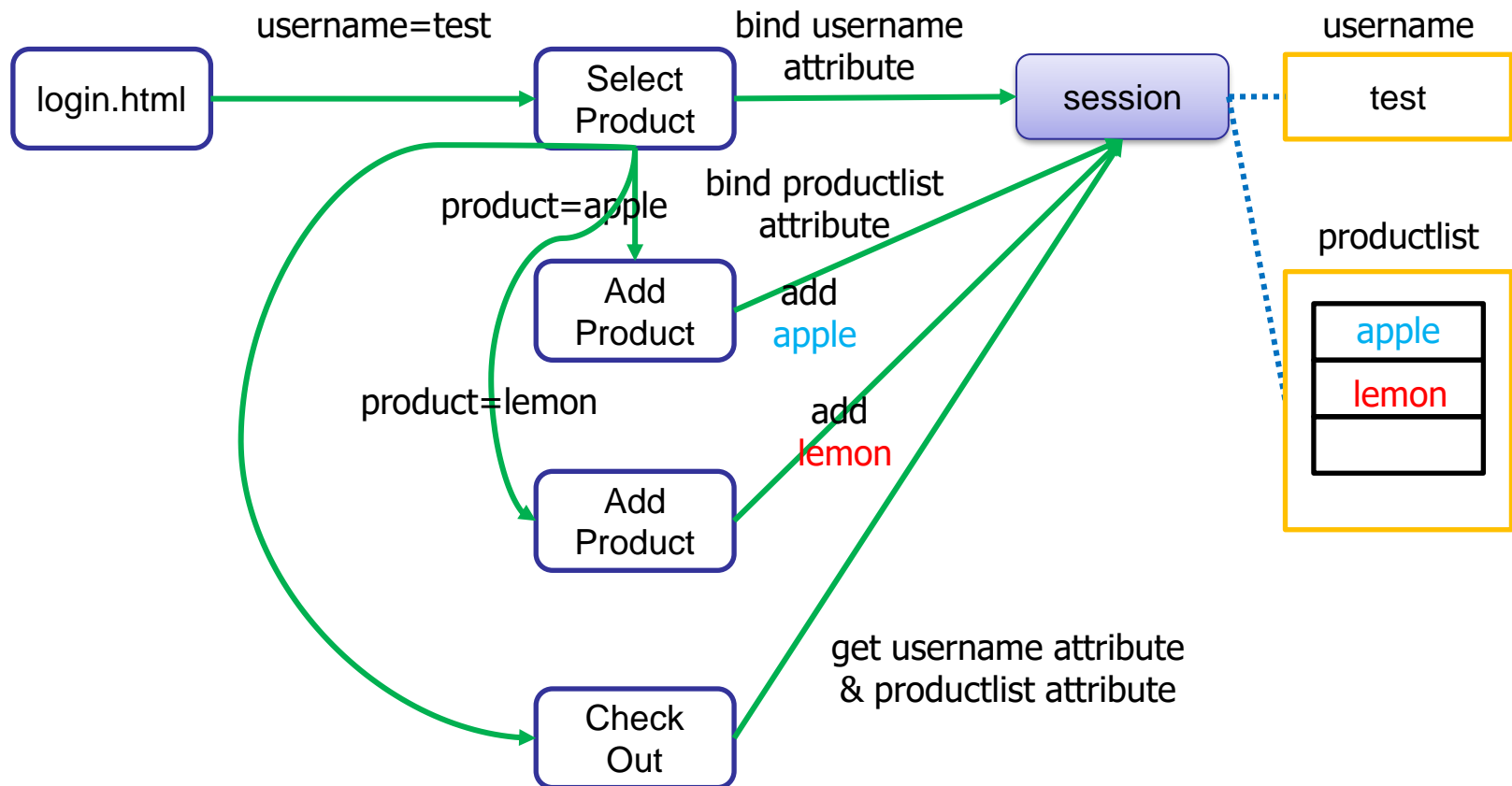
# 속성 (Attribute)



login.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>장바구니</title>
<style>
    body {text-align:center;}
</style>
</head>
<body>
    <H2> 로그인 </H2>
    <form name=form1 method=post action=/webservice/SelectProduct>
        <input type=text name=username />
        <input type=submit value=로그인 />
    </form>
</body>
</html>
```

# 속성 (Attribute) - HttpSession





# 속성 (Attribute)

SelectProduct.java

```
16 @WebServlet("/SelectProduct")
17 public class SelectProduct extends HttpServlet {
```

```
protected void doPost(HttpServletRequest request, HttpServletResponse response)
    response.setContentType("text/html; charset=UTF-8");
    request.setCharacterEncoding("UTF-8");
    String username = request.getParameter("username");
    HttpSession session = request.getSession();
    session.setAttribute("username", username);

    PrintWriter out = response.getWriter();
    out.println("<center>");
    out.println("<H2>상품 선택</H2>");
    out.println(username + "님이 로그인한 상태입니다");
    out.println("<form name=form2 method=post action=AddProduct>");
    out.println("    <select name=product>");
    out.println("        <option>apple</option>");
    out.println("        <option>orange</option>");
    out.println("        <option>lemon</option>");
    out.println("    </select>");
    out.println("    <input type=submit value=추가 />");
    out.println("</form>");
    out.println("<a href=CheckOut>결재</a>");
    out.println("</center>");
}
```

# 속성 (Attribute)

---

## AddProduct.java

```
17 @WebServlet("/AddProduct")
18 public class AddProduct extends HttpServlet {
```

```
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    String product = request.getParameter("product");
    HttpSession session = request.getSession();
    ArrayList<String> list = (ArrayList<String>)session.getAttribute("productlist");
    if (list == null)
        list = new ArrayList<String>();
    list.add(product);
    session.setAttribute("productlist", list);

    response.setContentType("text/html; charset=UTF-8");
    PrintWriter out = response.getWriter();
    out.println("<html><body>");
    out.println(product + "이(가) 추가되었습니다.");
    out.println("</body></html>");
}
```

# 속성 (Attribute)

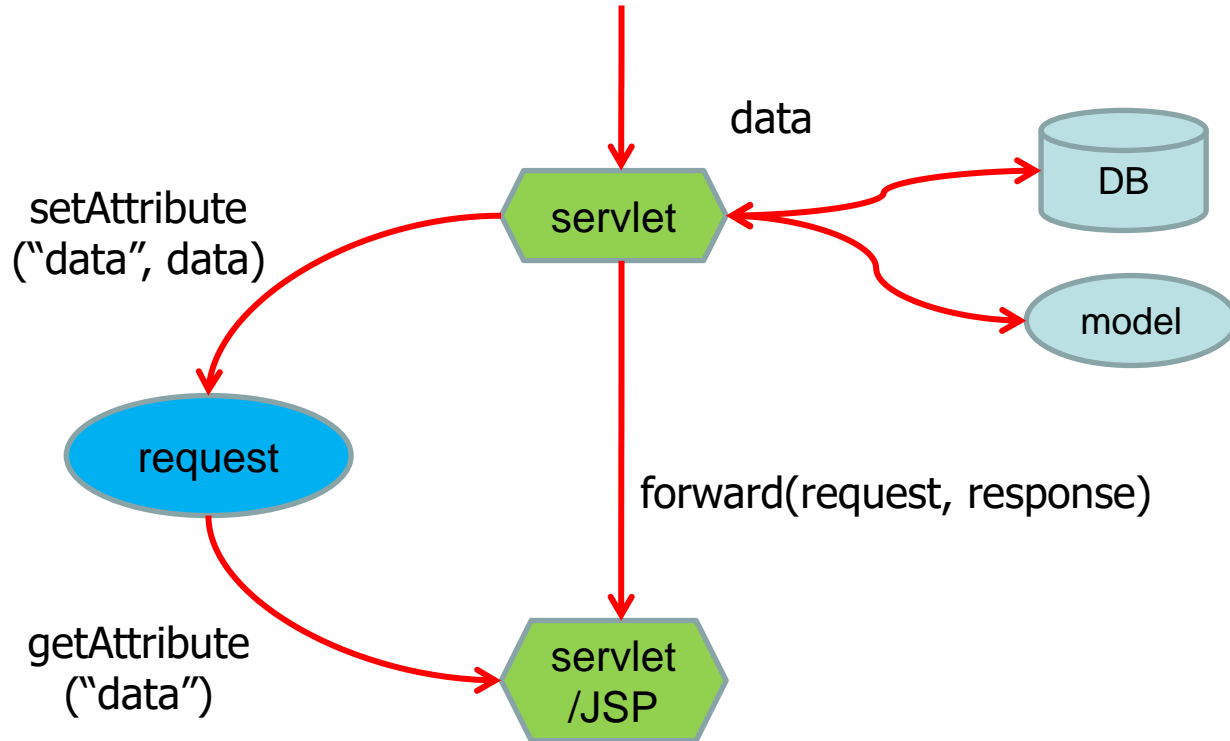
## CheckOut.java

```
17 @WebServlet("/CheckOut")  
18 public class CheckOut extends HttpServlet {
```

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
    response.setContentType("text/html; charset=UTF-8");  
    PrintWriter out = response.getWriter();  
  
    HttpSession session = request.getSession();  
    String name = (String)session.getAttribute("username");  
    ArrayList<?> list = (ArrayList<?>)session.getAttribute("productlist");  
  
    out.println("<html><body>");  
    out.println("<center> <H2>" + name+ "님이 선택한 상품 목록 </H2>");  
  
    if (list == null) {  
        out.println("선택한 상품이 없습니다!");  
    } else {  
        for (Object product:list) {  
            out.println(product + "<br>");  
        }  
        out.println("</center>");  
    }  
    out.println("</body></html>");  
}
```

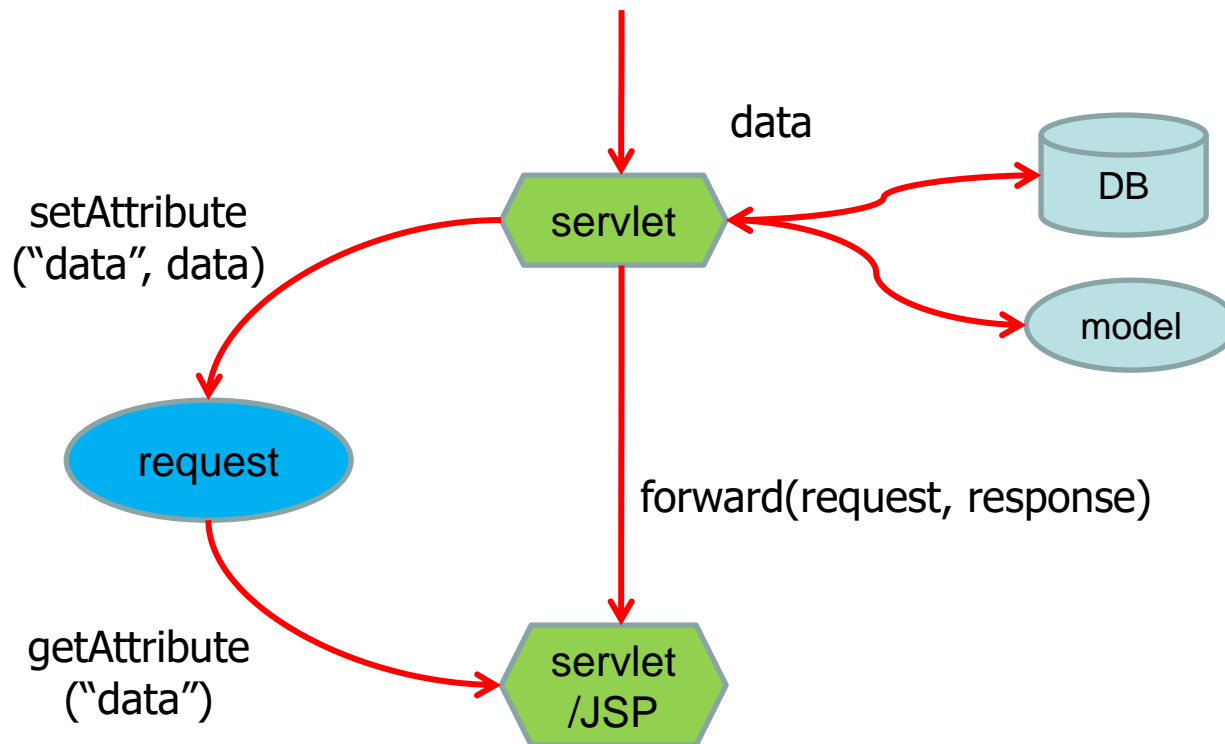
# 속성 (Attribute) - HttpServletRequest

- Request 객체와 속성
  - 한 애플리케이션 내의 다른 component에게 data를 넘겨주기 위해 사용
- Forwarding
  - RequestDispatcher 사용



# 속성 (Attribute)

- Request 객체와 속성
  - 한 애플리케이션 내의 다른 component에게 data를 넘겨주기 위해 사용
  - RequestDispatcher 사용



# 속성 (Attribute)

---

## Controller.java

```
protected void doPost(HttpServletRequest request, HttpServletResponse response)
    String username = request.getParameter("username");
    request.setAttribute("username", username);
    RequestDispatcher view = request.getRequestDispatcher("/View");
    view.forward(request, response);
}
```

## View.java

```
protected void doPost(HttpServletRequest request, HttpServletResponse
    response.setContentType("text/html; charset=UTF-8");
    PrintWriter out = response.getWriter();
    String username = (String)request.getAttribute("username");
    out.println("User name is " + username);
}
```