

# 4장. 서블릿 – Servlet

*Servlet API*



# JSP 페이지의 처리 과정

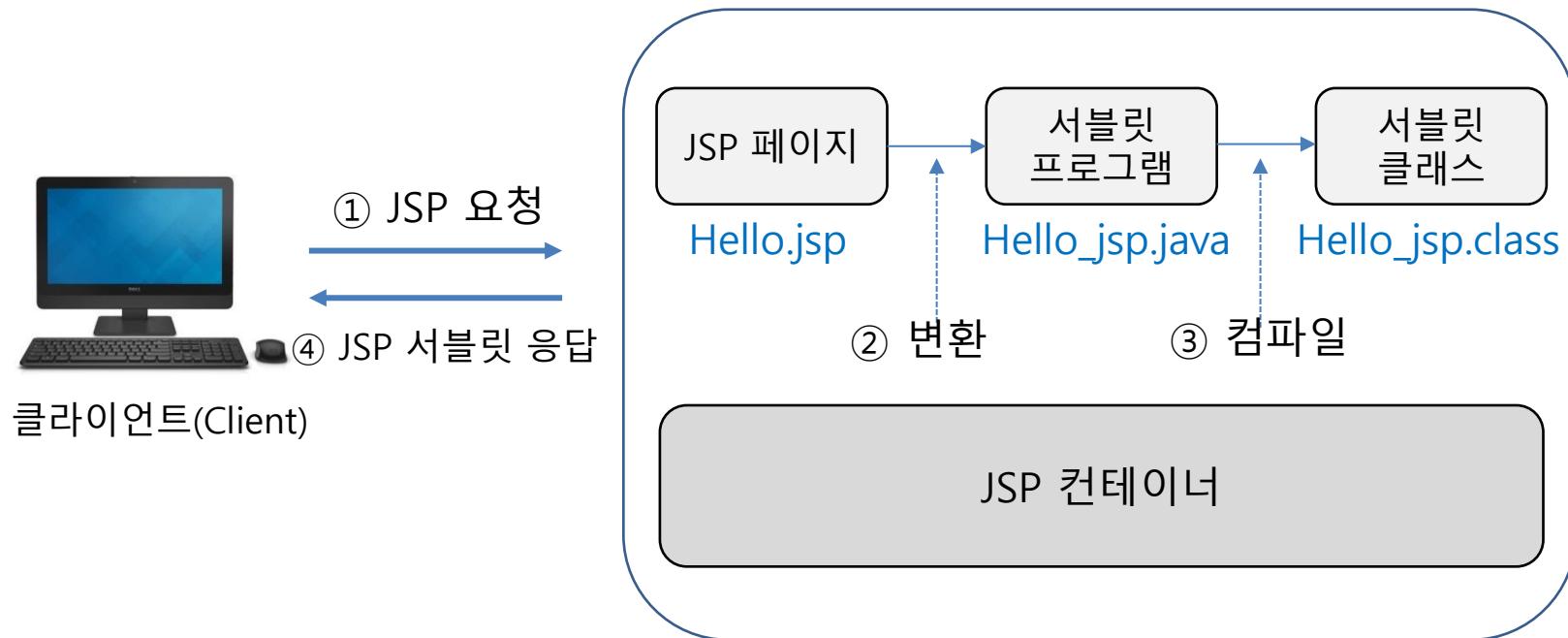
## ◆ JSP 페이지 처리 과정

정적인 HTML과 달리 JSP는 Java코드에 의해 동적인 페이지가 되었으므로, 그것을 컴파일할 서버가 필요해진다. 이것이 WAS(Web Application Server)인데, WAS는 정적인 HTML만 관리하던 웹서버와 달리, 동적인 페이지도 컴파일하는 것이 가능하다.

- ① 웹 브라우저가 웹 서버에 JSP를 요청한다. 웹 서버는 요청된 Hello.jsp에서 jsp 확장자 를 발견하여 웹 서버에 있는 JSP 컨테이너에 전달한다.
- ② JSP 컨테이너는 JSP 페이지를 서블릿 프로그램인 Hello\_jsp.java로 변환한다.
- ③ JSP 컨테이너가 서블릿 프로그램을 컴파일하여 Hello\_jsp.class로 만들고 이를 웹서버 에 전달한다.
- ④ 웹 서버는 정적 웹 페이지처럼 \*.class의 실행 결과를 웹 브라우저에 응답으로 전달하 므로 웹브라우저는 새로 가공된 HTML 페이지를 동적으로 처리한 결과를 보여 준다.



# JSP 페이지의 처리 과정



정적인 파일 – HTML이나 오브젝트는 웹 서버에서 처리.

동적인 파일 – 웹 컨테이너로 넘겨서 처리함



## JSP의 특징

- **JSP는 서블릿 기술의 확장이다.**

처음에는 서버 측 프로그래밍 방식으로 자바를 사용하는 서블릿(Servlet)을 먼저 개발하였으나 서블릿 개발방식이 쉽지 않아서 개발된 기술이 JSP(HTML코드에 삽입)이다

- **JSP는 유지 관리가 용이하다.**

서블릿 기술은 프레젠테이션(View)와 비즈니스 로직(Control)이 섞여 있지만 JSP는 분리할 수 있어서 관리가 쉽다.

- **JSP는 빠른 개발이 가능하다.**

코드를 수정했을때 서블릿에서는 다시 컴파일해야 하지만, JSP는 컴파일하지 않는다.

- **JSP로 개발하면 코드 길이를 줄일 수 있다.**

액션 태그, JSTL, 표현 언어를 사용하여 서블릿 보다 코드의 길이를 줄일 수 있다.



# 서블릿 – Servlet

## 서블릿 vs JSP

서블릿은 서버쪽에서 실행되면서 클라이언트의 요청에 따라 동적으로 서비스를 제공하는 **자바 클래스**이다.

서블릿의 문제점을 보완하여 등장한 것이 **JSP(Java Server Pages)**이다.

화면의 기능이나 구성이 복잡해짐에 따라 사용자를 고려하는 요구사항이 늘어났고 디자이너의 경우 화면의 수월한 기능 구현과 개발 후의 화면의 편리한 유지관리를 목적으로 도입 된 것이다.

현재는 웹 애플리케이션을 개발할 때 **JSP**는 화면 계층(프레젠테이션 계층)으로 서블릿은 비즈니스 로직(Controller)으로 역할을 나누어 기능을 구현하고 있다.**(모델2 방식- MVC 패턴)**



# Java EE API

## Java EE API-> 서블릿(servlet)

The screenshot shows a browser window displaying the Java EE 7 API documentation for the `HttpServlet` class. The URL is [docs.oracle.com/javaee/7/api/toc.htm](http://docs.oracle.com/javaee/7/api/toc.htm). The page is titled "HttpServlet (Java(TM) EE 7 Spec)" and has tabs for OVERVIEW, PACKAGE, CLASS (which is selected), USE, TREE, and DEPRECATED. Below the tabs are links for PREV CLASS and NEXT CLASS, and options for FRAMES or NO FRAMES. The main content area displays the `HttpServlet` class definition, its inheritance from `java.lang.Object` and `GenericServlet`, and its implementation of `Serializable`, `Servlet`, and `ServletConfig`. It also shows the abstract class definition:

```
public abstract class HttpServlet  
extends GenericServlet
```

The text explains that `HttpServlet` provides an abstract class to be subclassed to create an `Http` servlet, and that it must override at least one method, usually one of these:

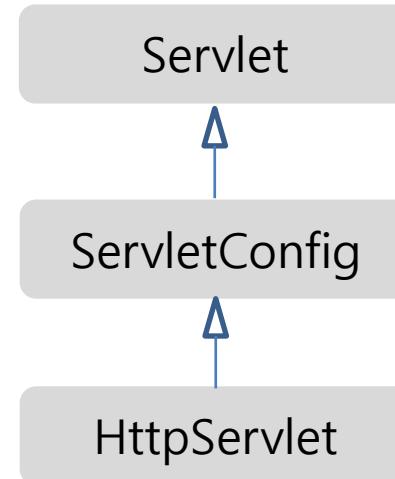
- `doGet`, if the servlet supports HTTP GET requests
- `doPost`, for HTTP POST requests

On the left side of the page, there is a sidebar with a tree view of Java packages, starting with `javax.security.auth.message` and listing various sub-packages and classes like `HttpSessionContext`, `HttpSessionIdListener`, etc. There are also sections for `HttpSessionListener`, `HttpUpgradeHandler`, `Part`, `WebConnection`, and a `Classes` section containing `Cookie`, `HttpServletRequest`, `HttpServletRequestWrapper`, and `HttpServletResponseWrapper`.



# Servlet API

## 서블릿 API 계층 구조

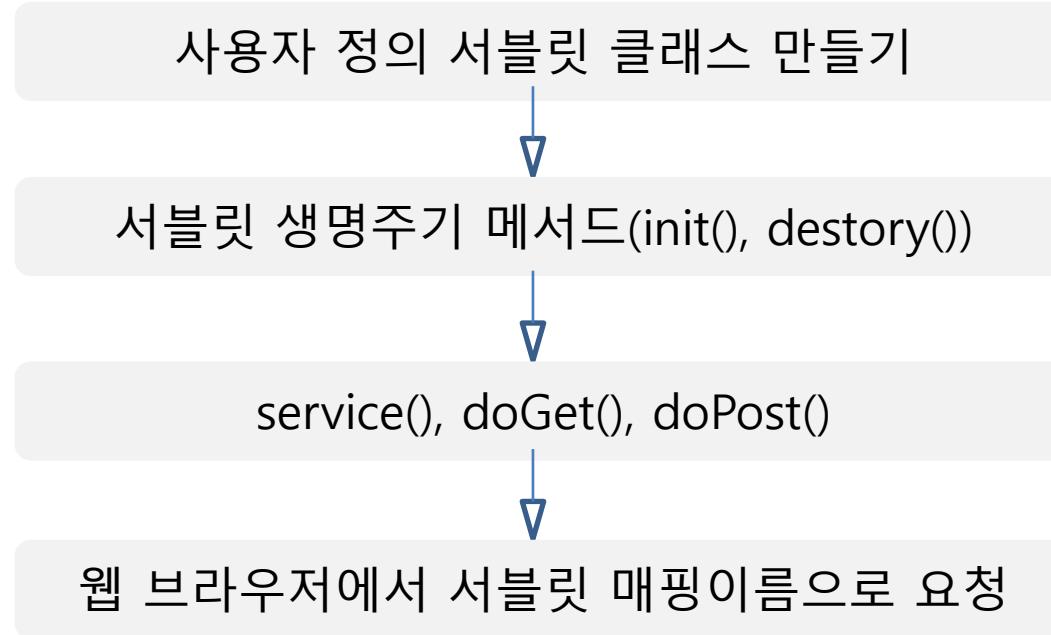


## HttpServlet의 주요 메서드의 기능

메서드	기능
<b>doGet</b> (HttpServletRequest req, HttpServletResponse resp)	서블릿이 GET request를 수행하기 위해 service()를 통해서 호출된다.
<b>doPost</b> (HttpServletRequest req, HttpServletResponse resp)	서블릿이 POST request를 수행하기 위해 service()를 통해서 호출된다.

# Servlet API

## 사용자 정의 서블릿 만들기



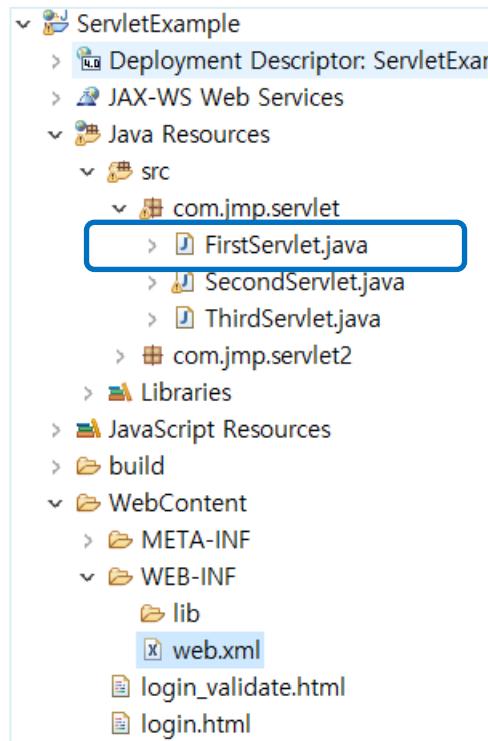
톰캣의 **servlet-api.jar** 클래스 패스 설정 - 버전이 낮은 경우 해당

프로젝트 이름 > 마우스 우측 > Build Path > Configure Build Path > Libraries  
탭 > Classpath > Add External JARs > **tomcat > lib > servlet-api.jar**

# 서블릿 만들기 - xml 설정으로 매팅하기

## FirstServlet 실습 - xml 설정으로 매팅하기

class 메뉴로 java 파일 만들기



```
public class FirstServlet extends HttpServlet{  
    private static final long serialVersionUID = 1L;
```

Override/Implement Methods

Enter method name, prefix or pattern (\*, ?, or camel case)

Search Methods

Select methods to override or implement:

- HttpServlet
  - doDelete(HttpServletRequest, HttpServletResponse)
  - doGet(HttpServletRequest, HttpServletResponse)
  - doHead(HttpServletRequest, HttpServletResponse)
  - doOptions(HttpServletRequest, HttpServletResponse)
  - doPost(HttpServletRequest, HttpServletResponse)
  - doPut(HttpServletRequest, HttpServletResponse)
  - doTrace(HttpServletRequest, HttpServletResponse)
  - getLastModified(HttpServletRequest)
  - service(HttpServletRequest, HttpServletResponse)
  - service(ServletRequest, ServletResponse)
- GenericServlet
  - destroy()
  - getInitParameter(String)
  - getInitParameterNames()
  - getServletConfig()

Override 한다.



# 서블릿 만들기 – xml 설정으로 매팅하기

## FirstServlet 실습

Java 코드 작성

```
public class FirstServlet extends HttpServlet{

    private static final long serialVersionUID = 1L;

    @Override
    public void init() throws ServletException {
        System.out.println("init() 호출");
    }

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse re
        System.out.println("doGet() 호출");
    }

    @Override
    public void destroy() {
        System.out.println("destroy() 호출");
    }
}
```



# 서블릿 만들기 – xml 설정으로 매팅하기

## FirstServlet 실습

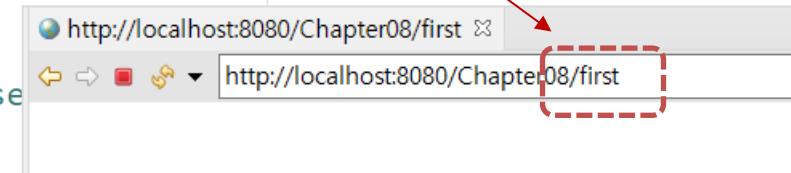
### 서블릿 매팅하기

1. 프로젝트에 있는 /WEB-INF/web.xml에 설정한다.
2. <servlet> 태그와 <servlet-mapping> 태그를 작성한다
3. 서버를 실행하고 주소창에 "http://localhost:8080/Chapter08/first"로 요청한다.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <servlet>
        <servlet-name>first</servlet-name>
        <servlet-class>comjmp.servlet.FirstServlet</servlet-class>
    </servlet>

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

웹 브라우저에 요청하는 매팅 이름



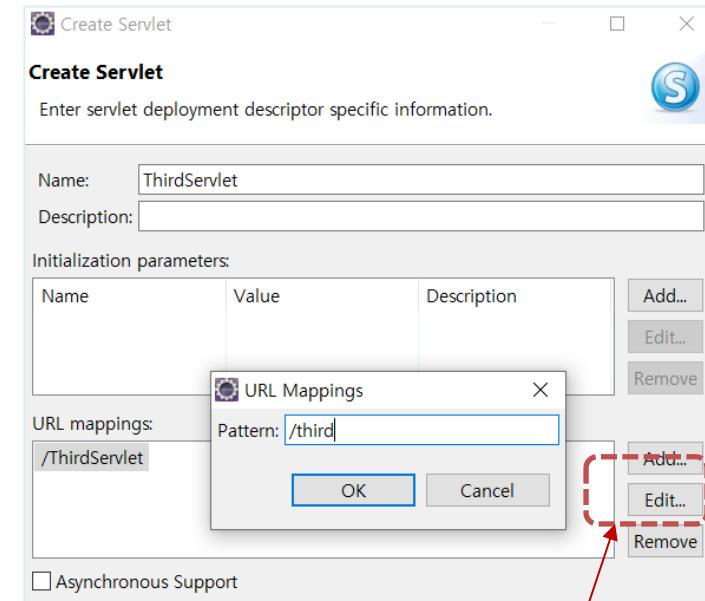
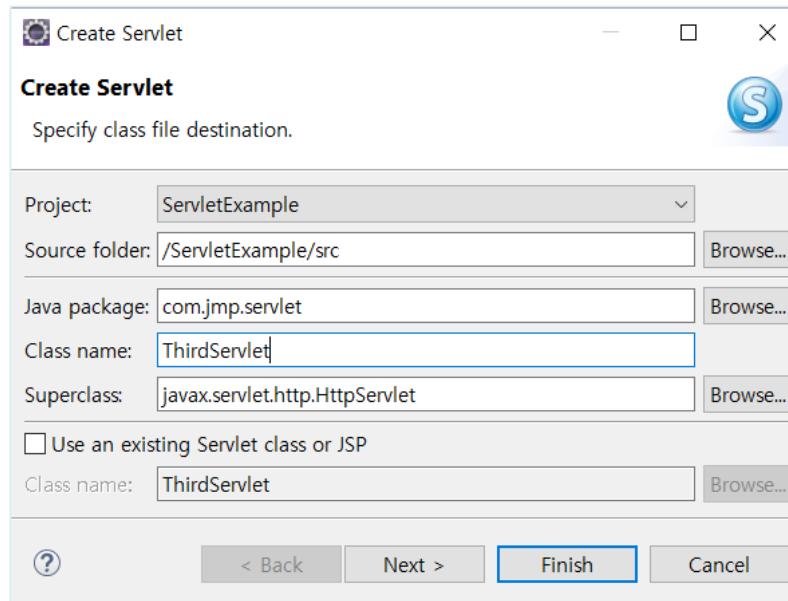
10월 19, 2020 6:35:36 오전 org.apache.catalina.core.ContainerBase.[CoyoteAdapter].INITORIGINATOR: 서버가 [1044] 밀리초 내에 시작되었습니다  
init() 호출  
doGet() 호출



# 서블릿 만들기 – 애너테이션으로 매팅하기

## ThirdServlet 실습 – 애너테이션(@)을 이용한 매팅 실습

- New > Servlet 실행

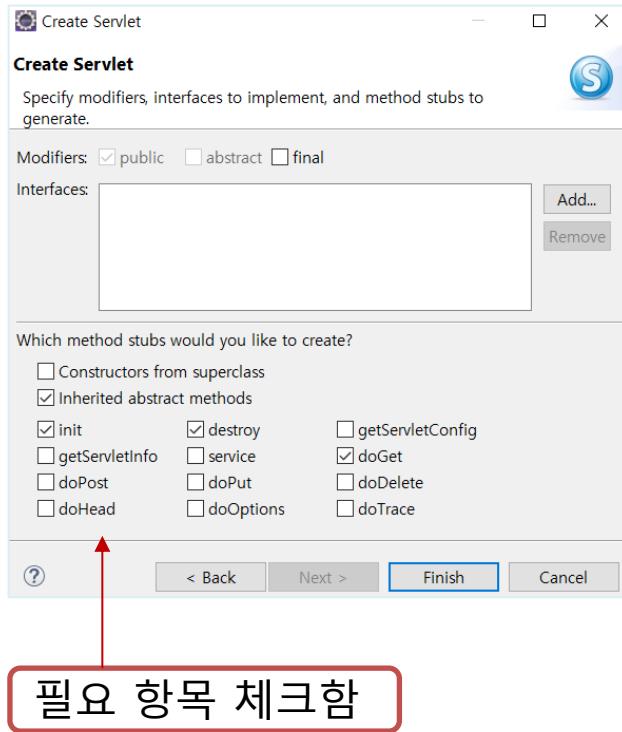


Edit > "/third"로 변경



# 서블릿 만들기 – 애너테이션으로 맵핑하기

## ThirdServlet 실습



웹 브라우저에 요청하는 맵핑 이름

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

    public void init(ServletConfig config) throws ServletException {
        System.out.println("init() 호출");
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) {
        System.out.println("doGet() 호출");
    }

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



# 서블릿의 응답 처리

## HttpServletResponse를 이용한 서블릿 응답 실습



### MIME-TYPE

톰캣 컨테이너에 미리 설정해 놓은 데이터 종류이다.

서버에서 웹 브라우저로 데이터를 전송할 때 데이터 종류를 지정해서 전송한다.



# 서블릿 만들기 – 애너테이션으로 매팅하기

## HelloServlet 실습

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

    @Override
    protected void service(HttpServletRequest req, HttpServletResponse resp)
        PrintWriter out = resp.getWriter();

        for(int i=0; i<10; i++)
            out.println((i+1)+ " : 안녕 Servlet!!");
    }

}
```

localhost:8181/Chapter09/hello

```
1 : ?? Servlet!!
2 : ?? Servlet!!
3 : ?? Servlet!!
4 : ?? Servlet!!
```

- ISO-8859-1 방식(웹서버 – 브라우저간) 1byte 전송
- UTF-8 방식 : 2byte 전송



# 서블릿 만들기 – 애너테이션으로 맵핑하기

## HelloServlet 실습 – 한글 콘텐츠 형식 출력하기

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

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
        response.setCharacterEncoding("utf-8");
        response.setContentType("text/html; charset=utf-8");

        PrintWriter out = response.getWriter();

        for(int i=0; i<10; i++) {
            out.println((i+1)+ " : 안녕 Servlet!!<br>");
        }
    }
}
```

1 : 안녕 Servlet!!  
2 : 안녕 Servlet!!  
3 : 안녕 Servlet!!  
4 : 안녕 Servlet!!  
5 : 안녕 Servlet!!



# 서블릿 만들기 – 애너테이션으로 맵핑하기

## HelloServlet 실습 – 한글 콘텐츠 형식 출력하기

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

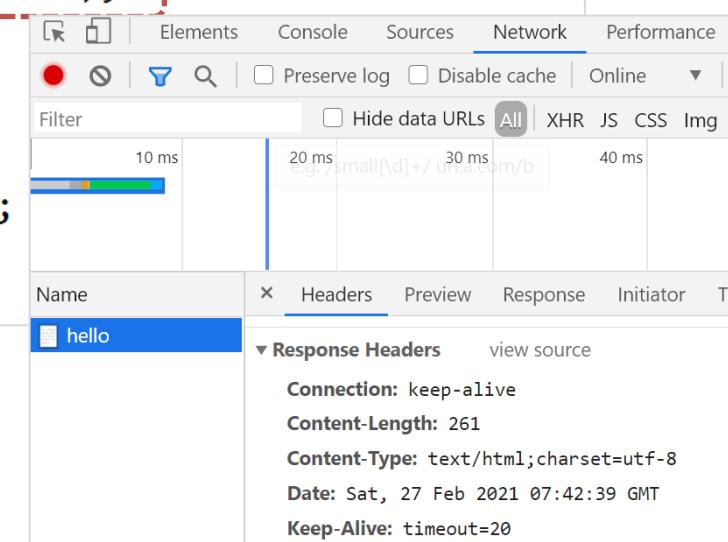
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
        response.setCharacterEncoding("utf-8");
        response.setContentType("text/html; charset=utf-8");
    }
}
```

```
PrintWriter out = response.getWriter();

for(int i=0; i<10; i++) {
    out.println((i+1)+ " : 안녕 Servlet!!<br>");
}
```

```
1 : 댐濮 Servlet!!<br>
2 : 댐濮 Servlet!!<br>
3 : 댐濮 Servlet!!<br>
4 : 댐濮 Servlet!!<br>
5 : 댐濮 Servlet!!<br>
```

```
1 : 안녕 Servlet!!
2 : 안녕 Servlet!!
3 : 안녕 Servlet!!
4 : 안녕 Servlet!!
5 : 안녕 Servlet!!
```



# 서블릿 만들기 – 애너테이션으로 맵핑하기

## GET 요청과 쿼리 스트링(입력)

http://localhost/hello

GET

http://localhost/hello?cnt=3

GET

안녕하세요  
안녕하세요  
안녕하세요

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

PrintWriter out = response.getWriter();

//int cnt = Integer.parseInt(request.getParameter("cnt"));

String param = request.getParameter("cnt"); //문자가 넘어옴
int cnt = 10;
if(param != null) { //NullPointerException 처리
    cnt = Integer.parseInt(param);
}

for(int i=0; i<cnt; i++) {
    out.println((i+1)+ " : 안녕 Servlet!!<br>");
}
```

← → ⌂ ⓘ localhost:8181/Chapter09/hello?cnt=4

1 : 안녕 Servlet!!  
2 : 안녕 Servlet!!  
3 : 안녕 Servlet!!  
4 : 안녕 Servlet!!



# 서블릿 만들기 – 애너테이션으로 맵핑하기

## GET 요청과 쿼리 스트링(입력)



localhost:8181/Chapter09/hello.jsp

인사 하기

인사 4번 하기

```
<body>
    <h2>쿼리 스트링(query string) 예제</h2>
    <p><a href="hello">인사 하기</a></p>

    <p><a href="hello?cnt=4">인사 4번 하기</a></p>
</body>
```

# <form> 태그로 서블릿에 요청하기

## HttpServletRequest로 요청 실습

메소드	반환유형	설명
getParameter(String name)	String	요청 파라미터 이름이 name인 값을 전달받음
getParameterValues(String name)	String[ ]	모든 요청 파라미터 이름이 name인 값을 배열 형태로 전달 받음
getParameterNames()	Java.util.Enumeration	모든 요청 파라미터의 이름과 값을 Enumeration 객체 타입으로 전달 받음

localhost:8080/Chapter08/servlet01.jsp

아이디 : corona

비밀번호 : ....

전송

localhost:8080/Chapter08/login

브라우저엔 내용이 출력되지 않음

2월 23, 2021 6:46:15 오전 org.apache.catalina.INFO: 서버가 [1188] 밀리초 내에 시작되었습니다.

아이디 : corona

비밀번호 : 2019

# <form> 태그로 서블릿에 요청하기

LoginServlet 맵핑 이름

```
<form action="Login" method="post">
    <p>
        <label for="id">아이디 : </label>
        <input type="text" id="id" name="userid">
    </p>
    <p>
        <label for="pwd">비밀번호 : </label>
        <input type="password" id="pwd" name="passwd">
    </p>
    <p><input type="submit" value="전송" onclick="checkForm()"></p>
</form>
```



# <form> 태그로 서블릿에 요청하기

## LoginServlet – 애너테이션 방식

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

    public void init(ServletConfig config) throws ServletException {
        System.out.println("init() 호출");
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response) {
        request.setCharacterEncoding("utf-8");
        String userid = request.getParameter("userid");
        String passwd = request.getParameter("passwd");

        System.out.println("아이디 : " + userid);
        System.out.println("비밀번호 : " + passwd);
    }
}
```



# 서블릿의 응답 처리 – html형식으로 응답

## LoginServlet2에서 웹 브라우저로 출력(보여주기)

The image shows two browser windows side-by-side. The left window, titled 'localhost:8080/Chapter08/servlet02.jsp', contains a form with fields for '아이디' (id) set to 'corona' and '비밀번호' (password) set to '....'. A button labeled '가입하기' (Join) is at the bottom. The right window, titled 'localhost:8080/Chapter08/login2', displays the submitted data: '아이디 : corona', '비밀번호 : 2019', and '전화번호 : 010-7979-3355'. A red arrow points from the left window to the right window, indicating the flow of data.

```
<form action="Login2" method="post">
    <p>
        <label for="id">아이디 : </label>
        <input type="text" id="id" name="userid">
    </p>
    <p>
        <label for="pwd">비밀번호 : </label>
        <input type="password" id="pwd" name="passwd">
    </p>
    <p><input type="hidden" name="phone" value="010-7979-3355"></p>
    <p><input type="submit" value="가입하기"></p>
</form>
```

hidden 태입으로 정보 보내기

# 서블릿의 응답 처리 – html형식으로 응답

```
@WebServlet("/login2")
public class LoginServlet2 extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doPost(HttpServletRequest request, HttpServletResponse response) {
        request.setCharacterEncoding("utf-8");
        response.setContentType("text/html; charset=utf-8");
        //웹 브라우저로 응답을 html형식으로 설정

        PrintWriter out = response.getWriter();
        //브라우저에 출력할 out 객체 생성

        String id = request.getParameter("userid");
        String pwd = request.getParameter("passwd");
        String phone = request.getParameter("phone");

        String data = "<html><body>";
        data += "<p>아이디 : " + id;
        data += "<p>비밀번호 : " + pwd;
        data += "<p>전화번호 : " + phone;
        data += "</body></html>";

        out.println(data);
    }
}
```



# <form> 태그로 서블릿에 요청하기

## RegisterServlet 생성 – 여러 개의 name을 전송할 때의 요청 처리

```
<form action="register" method="post">
    <p>
        <label for="id">아이디 : </label>
        <input type="text" id="id" name="userid">
    </p>
    <p>
        <label for="pwd">비밀번호 : </label>
        <input type="password" id="pwd" name="passwd">
    </p>
    <p>
        <label for="subj">과목 : </label>
        <input type="checkbox" name="subject" value="java" checked>java
        <input type="checkbox" name="subject" value="jsp" >jsp
        <input type="checkbox" name="subject" value="spring" >spring
        <input type="checkbox" name="subject" value="android" >android
    </p>
    <p>
        <input type="submit" value="신청하기">
        <input type="reset" value="다시입력">
    </p>
</form>
```

아이디 : ai2021  
비밀번호 :   
과목 : java jsp spring android

# <form> 태그로 서블릿에 요청하기

## RegisterServlet 생성

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

    public void init(ServletConfig config) throws ServletException {
        System.out.println("init() 호출");
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) {
        response.setCharacterEncoding("utf-8");

        String userid = request.getParameter("userid");
        String passwd = request.getParameter("passwd");
        String[] subject = request.getParameterValues("subject");

        System.out.println("아이디 : " + userid);
        System.out.println("비밀번호 : " + passwd);
        for(String subj : subject)
            System.out.println("선택한 과목 : " + subj);
    }
}
```

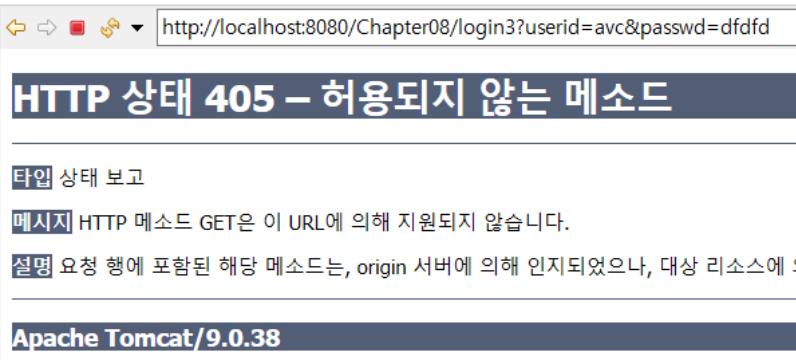
2월 23, 2021 7:12:33 오후  
INFO: 서버가 [1338] 밀리초  
아이디 : ai2021  
비밀번호 : 2021  
선택한 과목 : java  
선택한 과목 : jsp  
선택한 과목 : spring



# GET방식과 POST 방식으로 동시 처리

## GET방식과 POST 방식 요청 동시에 처리하기

```
<form action="Login3" method="get">  
    <p>  
        <label for="id">아이디 : </label>  
        <input type="text" id="id" name="userid">  
    </p>  
    <p>  
        <label for="pwd">비밀번호 : </label>  
        <input type="password" id="pwd" name="passwd">  
    </p>  
    <p><input type="submit" value="가입하기"></p>  
</form>
```



12월 01, 2020 1:00:29 오후  
정보: 서버가 [393] 밀리초 내어  
doGet() 호출  
아이디 : myuser  
비밀번호 : 1111  
전화번호 : 010-1234-5678

# GET방식과 POST 방식으로 동시 처리

## GET방식과 POST 방식 요청 동시에 처리하기

### - 방법 1

```
@WebServlet("/login3")
public class LoginServlet3 extends HttpServlet {

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        System.out.println("doGet() 호출");
        doHandle(request, response);
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        System.out.println("doPost() 호출");
        doHandle(request, response);
    }

    protected void doHandle(HttpServletRequest request, HttpServletResponse response) {
        request.setCharacterEncoding("utf-8");
        String userid = request.getParameter("userid");
        String passwd = request.getParameter("passwd");
        String phone = request.getParameter("phone");

        System.out.println("아이디 : " + userid);
        System.out.println("비밀번호 : " + passwd);
        System.out.println("전화번호 : " + phone);
    }
}
```



# GET방식과 POST 방식으로 동시 처리

## GET방식과 POST 방식 요청 동시에 처리하기 - 간단

### - 방법 2

```
@WebServlet("/login3")
public class LoginServlet3 extends HttpServlet {
    private static final long serialVersionUID = 1L;

    public void init(ServletConfig config) throws ServletException {
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        doPost(request, response);
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        request.setCharacterEncoding("utf-8");
        String userid = request.getParameter("userid");
        String passwd = request.getParameter("passwd");
        String phone = request.getParameter("phone");

        System.out.println("아이디 : " + userid);
        System.out.println("비밀번호 : " + passwd);
        System.out.println("전화번호 : " + phone);
```



# 서블릿 포워드 기능

## 포워드(forward) 기능

- 하나의 서블릿에서 다른 서블릿이나 JSP와 연동하는 방법

## 포워드(forward) 기능이 사용되는 용도

- 요청(request)에 대한 추가 작업을 다른 서블릿에게 수행하게 함
- 요청에 대한 정보를 포함시켜 다른 서블릿에게 전달할 수 있음

## 포워드 방법

1. redirect 방법 – HttpServletResponse 객체의 sendRedirect() 메서드 이용
2. Refresh 방법 - HttpServletResponse 객체의 addHeader() 메서드 이용
3. location 방법 – 자바스크립트 location 객체의 href 속성을 이용
4. dpatch 방법 – RequestDispatcher 클래스의 forward() 메서드 이용

1,2,3은 브라우저가 재요청하는 방식이고, 4는 서블릿이 직접 요청하는 방식



# 서블릿 포워드 기능

## 1.redirect 방법 실습



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

    protected void doGet(HttpServletRequest request, HttpServletResponse response) {
        response.setContentType("text/html; charset=utf-8");

        response.sendRedirect("second");
    }
}
```

# 서블릿 포워드 기능

## 1. redirect 방법 실습

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

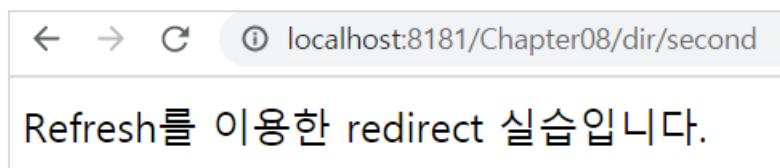
    protected void doGet(HttpServletRequest request, HttpServletResponse response) {
        response.setContentType("text/html; charset=utf-8");
        PrintWriter out = response.getWriter();

        out.println("<html><body>");
        out.println("sendRedirect를 이용한 redirect 실습입니다.");
        out.println("</body></html>");
    }
}
```



# 서블릿 포워드 기능

## 2. Refresh 방법 실습



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

    protected void doGet(HttpServletRequest request, HttpServletResponse response) {
        response.setContentType("text/html; charset=utf-8");
        PrintWriter out = response.getWriter();
        response.addHeader("Refresh", "1; url=second");
        //1초후 second로 이동
    }
}
```

# 서블릿 포워드 기능

## 2. Refresh 방법 실습

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

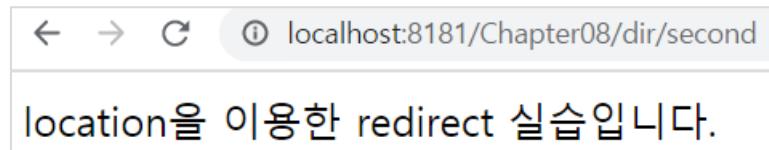
    protected void doGet(HttpServletRequest request, HttpServletResponse response) {
        response.setContentType("text/html; charset=utf-8");
        PrintWriter out = response.getWriter();

        out.println("<html><body>");
        out.println("Refresh를 이용한 redirect 실습입니다.");
        out.println("</body></html>");
    }
}
```



# 서블릿 포워드 기능

## 3. location방법 실습



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

    protected void doGet(HttpServletRequest request, HttpServletResponse response) {
        response.setContentType("text/html; charset=utf-8");
        PrintWriter out = response.getWriter();

        out.println("<script>");
        out.println("location.href='second'");
        out.println("</script>");
    }
}
```

# 서블릿 포워드 기능

## 3. location방법 실습

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

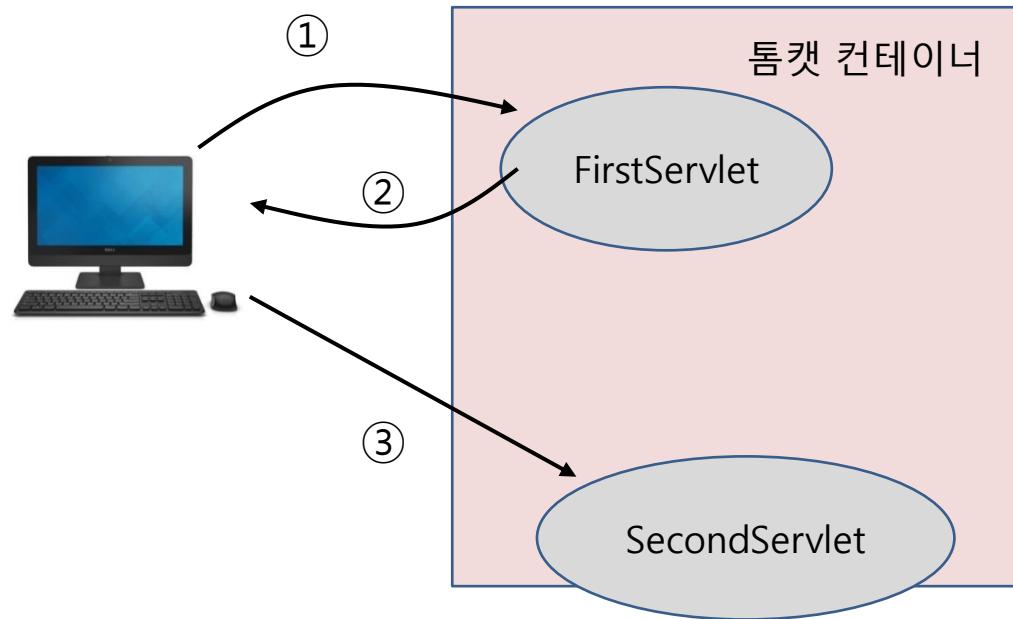
    protected void doGet(HttpServletRequest request, HttpServletResponse response) {
        response.setContentType("text/html; charset=utf-8");
        PrintWriter out = response.getWriter();

        out.println("<html><body>");
        out.println("location을 이용한 redirect 실습입니다.");
        out.println("</body></html>");
    }
}
```



# 서블릿 포워드 기능

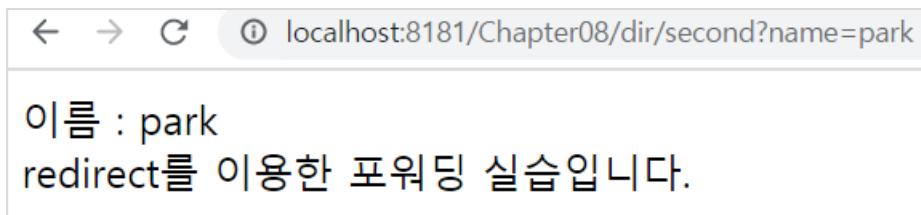
## redirect를 이용한 포워딩 과정



- ① 클라이언트의 웹 브라우저에서 첫 번째 서블릿에 요청.
- ② 첫 번째 서블릿은 `sendRedirect()` 메서드를 이용해 웹 브라우저에게 다시 요청하게 함
- ③ 클라이언트 웹 브라우저는 두 번째 서블릿으로 다시 요청

# 서블릿 포워드 기능

## 1-2. redirect로 데이터 전달하기



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

    protected void doGet(HttpServletRequest request, HttpServletResponse response) {
        response.setContentType("text/html; charset=utf-8");

        response.sendRedirect("second?name=park");
        //get방식으로 데이터 전달하기
    }
}
```

# 서블릿 포워드 기능

## 1-2. redirect로 데이터 전달하기

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

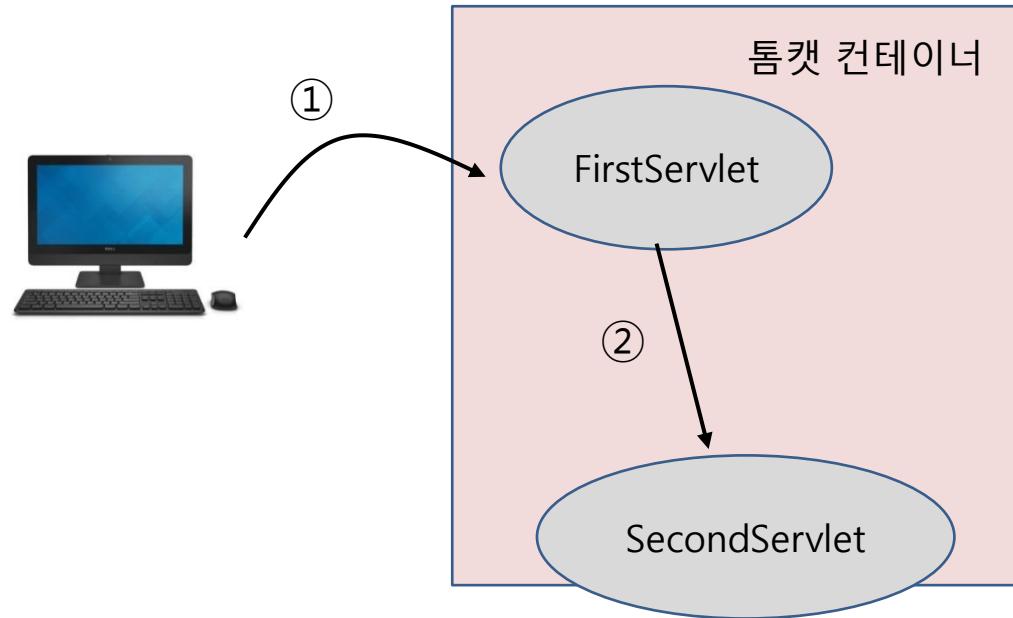
    protected void doGet(HttpServletRequest request, HttpServletResponse response) {
        response.setContentType("text/html; charset=utf-8");
        PrintWriter out = response.getWriter();

        String name = request.getParameter("name");
        out.println("<html><body>");
        out.println("이름 : " + name + "<br>");
        out.println("redirect를 이용한 포워딩 실습입니다.");
        out.println("</body></html>");
    }
}
```



# 서블릿 포워드 기능

## dispatch를 이용한 포워딩 과정



- ① 클라이언트의 웹 브라우저에서 첫 번째 서블릿에 요청합니다.
- ② 첫 번째 서블릿은 `RequestDispatcher`를 이용해 두 번째 서블릿으로 포워드합니다.

# 서블릿 포워드 기능

Java EE API-> 서블릿 -> RequestDispatcher 클래스

```
javax.security.auth.message
javax.security.jacc
javax.servlet
javax.servlet.annotation
javax.servlet.descriptor
javax.servlet.http
javax.servlet.jsp
javax.servlet.jsp.el
javax.servlet.jsp.tagext.core
ReadListener
Registration
Registration.Dynamic
RequestDispatcher
Servlet
ServletConfig
ServletContainerInitializer
```

javax.servlet

## Interface RequestDispatcher

```
public interface RequestDispatcher
```

Defines an object that receives requests from the client and sends them on the server. The servlet container creates the RequestDispatcher object located at a particular path or given by a particular name.

This interface is intended to wrap servlets, but a servlet container can wrap any resource.

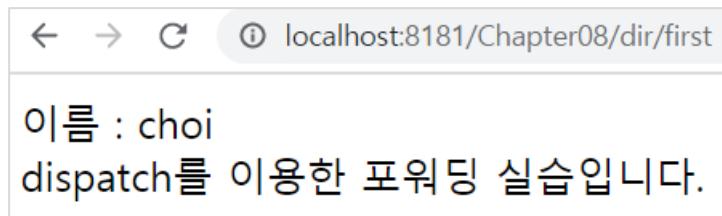
클라이언트에서 요청을 수신하여 서버의 리소스(예: 서블릿, HTML 파일 또는 JSP 파일)로 보내는 객체를 정의합니다.

서블릿 컨테이너는 RequestDispatcher 객체를 생성하며, 이 객체는 특정 경로에 있거나 특정 이름에 의해 지정된 서버 리소스를 감싸는 래퍼로 사용됩니다.



# 서블릿 포워드 기능

## 4. dispatch방법 실습



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

    protected void doGet(HttpServletRequest request, HttpServletResponse response) {
        response.setContentType("text/html; charset=utf-8");
        RequestDispatcher dispatch =
            request.getRequestDispatcher("second?name=choi");
        dispatch.forward(request, response);
    }
}
```

# 서블릿 포워드 기능

## 4. dispatch방법 실습

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

    protected void doGet(HttpServletRequest request, HttpServletResponse response) {
        response.setContentType("text/html; charset=utf-8");
        PrintWriter out = response.getWriter();
        String name = request.getParameter("name");
        out.println("<html><body>");
        out.println("이름 : " + name + "<br>");
        out.println("dispatch를 이용한 포워딩 실습입니다.");
        out.println("</body></html>");
    }
}
```

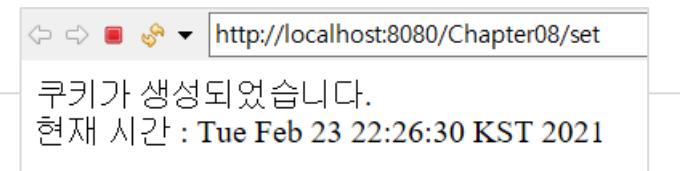


# 쿠키(cookie)

## 서블릿에서 쿠키 사용하기

- SetCookie 클래스.

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws Se  
response.setContentType("text/html; charset=utf-8");  
Date date = new Date();  
  
PrintWriter out = response.getWriter();  
Cookie cookie = new Cookie("cookieTest", URLEncoder.encode("JSP프로그래밍", "utf-8"));  
                                //아스키 코드로 저장  
cookie.setMaxAge(24*60*60);    //유효 기간 - 1일로 설정  
response.addCookie(cookie);    //생성된 쿠키를 브라우저로 전송  
  
out.println("쿠키가 생성되었습니다.<br>");  
out.println("현재 시간 : " + date);  
}
```



# 쿠키(cookie)

## 서블릿에서 쿠키 사용하기

- GetCookie 클래스

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException {
    response.setContentType("text/html; charset=utf-8");
    PrintWriter out = response.getWriter();

    Cookie[] cookies = request.getCookies();
    for(int i=0; i<cookies.length; i++) {
        boolean name = cookies[i].getName().equals("cookieTest"); //쿠키 이름이 일치하면
        if(name){
            String value = URLDecoder.decode(cookies[i].getValue(), "utf-8"); //쿠키값 디코딩.
            out.println("<h3>쿠키 값 가져오기 " + value + "</h3>");
        }
    }
}
```



Name	Value	D...	P...	Ex...
cookieTest	JSP%ED%94%84%EB%A...	lo...	/...	20...

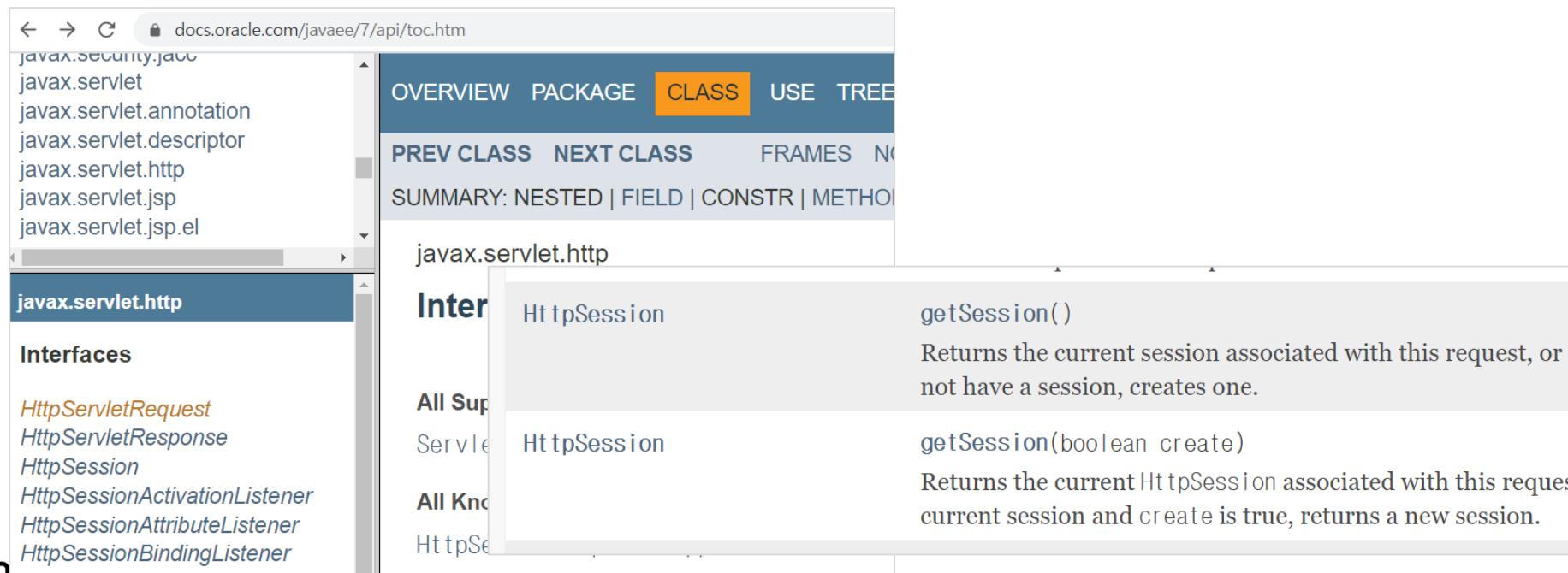
# 세션(session)

## 서블릿에서 세션 사용하기

HttpSession – Java EE API로 검색

서블릿에서 세션을 이용하려면 HttpSession 인터페이스의 객체를 생성해야 한다.

HttpSession 객체는 HttpServletRequest의 getSession() 메서드를 호출해서 생성한다.



The screenshot shows the Java EE 7 API documentation for the `javax.servlet.http` package. The `HttpSession` interface is the current focus. The `getSession()` method is highlighted with a yellow background, indicating it is the key method for obtaining a session object from a request.

`getSession()`  
Returns the current session associated with this request, or  
not have a session, creates one.

`getSession(boolean create)`  
Returns the current HttpSession associated with this request.  
current session and create is true, returns a new session.

# 세션(session)

## 서블릿에서 세션 사용하기

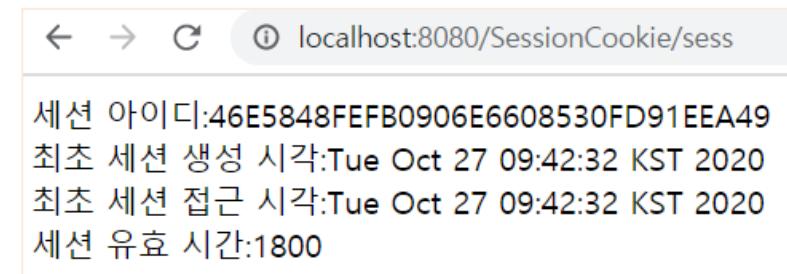
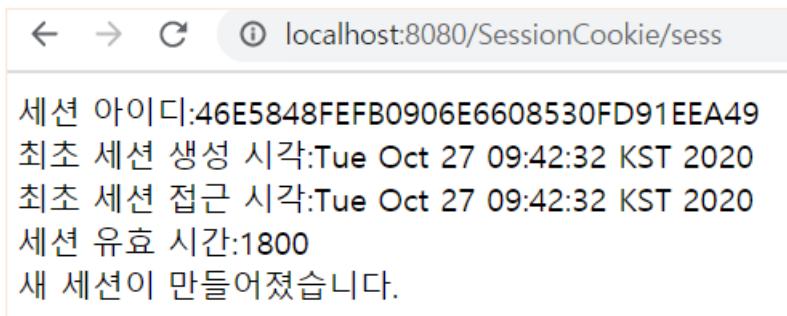
```
@WebServlet("/sess")
public class SessionServlet extends HttpServlet{
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
        response.setContentType("text/html; charset=utf-8");
    PrintWriter out = response.getWriter();

    //getSession() 을 호출하여 세션이 없으면 새로 생성하고, 세션이 있으면 기존 세션을 가져온다.
    HttpSession session = request.getSession();
    out.println("세션 아이디:" + session.getId() + "<br>");
    out.println("최초 세션 생성 시각:" + new Date(session.getCreationTime()) + "<br>");
    out.println("최초 세션 접근 시각:" + new Date(session.getLastAccessedTime()) + "<br>");
    out.println("세션 유효 시간:" + session.getMaxInactiveInterval() + "<br>");
    if(session.isNew()) { //최초 생성된 세션인지 판별함.
        out.print("새 세션이 만들어졌습니다.");
    }
}
```



# 세션(session)

## 서블릿에서 세션 사용하기



같은 브라우저에서 다른 탭으로 요청  
“새 세션이 만들어졌습니다.” 출력 안됨

Elements Console Sources Network Performance Memory Application Security L

Filter Only show cookies with an

Name	Value	Do...	Path	Expi...	Size
notShowPopfalse	notShowPopfalse	loca...	/	Ses...	1
notShowPop	true	loca...	/	202...	1
JSESSIONID	46E5848FEFB0906E6608530FD91E...	loca...	/Ses...	Ses...	4
cookieTest	JSP%ED%94%84%EB%AT%9C%	Toca...	/Ses...	202...	8

Select a cookie to preview its value

브라우저에 저장된 세션 쿠키

# 세션(session)

## 서블릿에서 세션 사용 실습

로그인

아이디	<input type="text" value="korea"/>
비밀번호	<input type="password"/>
<input type="button" value="로그인"/> <input type="button" value="취소"/>	

localhost:8181/Chapter09/sess

korea 님이 로그인했습니다.

```
<h2>로그인</h2>
<hr>
<form action="sess" method="post">
    <table>
        <tr>
            <td>아이디</td>
            <td><input type="text" name="id"></td>
        </tr>
        <tr>
            <td>비밀번호</td>
            <td><input type="password" name="passwd"></td>
        </tr>
        <tr>
            <td colspan="2">
                <input type="submit" value="로그인"/>
                <input type="reset" value="취소"/>
            </td>
        </tr>
    </table>
</form>
```



# 세션(session)

## 서블릿에서 세션 사용하기

```
@WebServlet("/sess")
public class LoginServlet extends HttpServlet{
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        response.setContentType("text/html; charset=utf-8");
        PrintWriter out = response.getWriter();

        //폼 입력 자료 수집
        String id = request.getParameter("id");
        String pwd = request.getParameter("passwd");

        //세션 클래스 사용
        HttpSession session = request.getSession();

        if(id.equals("korea") && pwd.equals("2021")) {
            session.setAttribute("userId", id); //세션 발급
            out.println(id + "님이 로그인했습니다.");
        }else {
            out.println("<script>");
            out.println("alert('아이디나 비밀번호가 틀립니다.')");
            out.println("history.go(-1)");
            out.println("</script>");
        }
}
```



# 세션(session)

## 서블릿에서 세션 사용하기 – 세션 삭제시

The image shows two side-by-side browser windows. Both windows have the URL `localhost:8080/SessionCookie/sess2`.  
The left window displays session information:

- 세션 아이디:27DC273708DD64F3EA606C9B4B80CC3F
- 최초 세션 생성 시각:Tue Oct 27 10:30:55 KST 2020
- 최초 세션 접근 시각:Tue Oct 27 10:30:55 KST 2020
- 세션 유효 시간:1800
- 새 세션이 만들어졌습니다.

A red arrow points from the session ID in the first window to the session ID in the second window.  
The right window displays session information:

- 세션 아이디:C7D545EC0B3AC5F66F3A2DCB8670B7BC
- 최초 세션 생성 시각:Tue Oct 27 10:31:14 KST 2020
- 최초 세션 접근 시각:Tue Oct 27 10:31:14 KST 2020
- 세션 유효 시간:1800
- 새 세션이 만들어졌습니다.

재요청시 세션 아이디가 바뀜

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws S
    response.setContentType("text/html; charset=utf-8");
    PrintWriter out = response.getWriter();

    //getSession() 을 호출하여 세션이 없으면 새로 생성하고, 세션이 있으면 기존 세션을 가져온다.
    HttpSession session = request.getSession();
    out.println("세션 아이디:" + session.getId() + "<br>");
    out.println("최초 세션 생성 시각:" + new Date(session.getCreationTime()) + "<br>");
    out.println("최초 세션 접근 시각:" + new Date(session.getLastAccessedTime()) + "<br>");
    out.println("세션 유효 시간:" + session.getMaxInactiveInterval() + "<br>");
    if(session.isNew()) { //최초 생성된 세션인지 판별함.
        out.print("새 세션이 만들어졌습니다.");
    }
}
session.invalidate();
```



# 세션(session)

## 서블릿에서 세션 사용하기 – 세션 유효 시간 변경하기

```
← → ⌂ ① localhost:8080/SessionCookie/sess3  
세션 아이디:F368DF99A3E539E0A4B6B466298D7EFE  
최초 세션 생성 시각:Tue Oct 27 10:48:26 KST 2020  
최초 세션 접근 시각:Tue Oct 27 10:48:26 KST 2020  
기본 세션 유효 시간:1800  
세션 유효 시간:5  
새 세션이 만들어졌습니다.
```

은행사이트에 로그인한 경우 10분  
에서 초단위로 역카운팅 되면서  
10분이 지나면 자동로그아웃 됨

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws  
    response.setContentType("text/html; charset=utf-8");  
    PrintWriter out = response.getWriter();  
  
    //getSession() 을 호출하여 세션이 없으면 새로 생성하고, 세션이 있으면 기존 세션을 가져온다.  
    HttpSession session = request.getSession();  
    out.println("세션 아이디:" + session.getId() + "<br>");  
    out.println("최초 세션 생성 시각:" + new Date(session.getCreationTime()) + "<br>");  
    out.println("최초 세션 접근 시각:" + new Date(session.getLastAccessedTime()) + "<br>");  
    out.println("기본 세션 유효 시간:" + session.getMaxInactiveInterval() + "<br>");  
    session.setMaxInactiveInterval(5); //세션 유효시간을 5초로 설정  
    out.println("세션 유효 시간:" + session.getMaxInactiveInterval() + "<br>");  
    if(session.isNew()) { //최초 생성된 세션인지 판별함.  
        out.print("새 세션이 만들어졌습니다.");  
    }  
}
```

