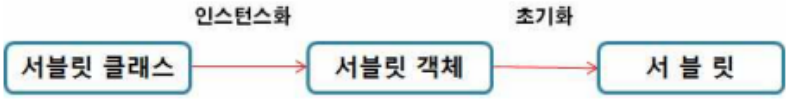
[02-02] JSP Day-3

1. 서블릿(Servlet)

서블릿 구현을 위해서는 HttpServlet 클래스를 상속한 서블릿 클래스를 작성해야 한다. 이 서블 릿 클래스는 컴파일 된 후 톰캣(Tomcat)과 같은 서블릿 컨테이너(웹 컨테이너, WAS – Web Application Server라고도 함) 안에서 서블릿 객체로 생성되어 실행된다. 서블릿 컨테이너는 서블 릿 클래스를 서블릿 객체로 만든 다음 그 객체를 초기화하여 요청을 처리하게 된다.



서블릿 컨테이너는 여러 사용자의 동시 요청을 처리하기 위해 하나의 서블릿 클래스를 가지고 멀티 스레딩 방식으로 서블릿을 운영한다. JSP 페이지와 서블릿 클래스는 서블릿 컨테이너 안에서 모두 서블릿으로 동작되며 상호 연계되어 사용된다. 또한 MVC(Model View Controller) 패턴에서 JSP는 뷰 페이지를 담당하고 서블릿은 컨트롤러 역할을 담당한다.

2. HTTP와 서블릿의 동작원리

2.1 HTTP(HyperText Transfer Protocol)

우리가 웹 브라우저를 통해 인터넷 서핑을 할 때 기본적으로 사용하는 프로토콜이 바로 HTTP 이 다. 이 HTTP는 웹 브라우저를 통해서 웹 서버와 통신하여 HTML 문서를 볼 수 있게 해 주는 프로 토콜로 비 연결(Connectionless)과 무 상태(Stateless) 속성을 가지고 있다. 비 연결이란 클라이언 트의 요청에 응답한 후 바로 연결을 끊어 다음 페이지로 연결이 유지되지 않는 것을 의미하고 무 상태란 서비스를 요청한 사용자가 누구인지 서버가 모르는 상태를 의미한다. 다시 말해 클라이언트 의 요청이 들어오면 서버는 그 요청을 처리하고 그 결과를 웹 브라우저에 전송한 후 클라이언트와 의 연결을 바로 끊어 버리기 때문에 사용자가 다음 페이지로 이동 할 때 이전 페이지에서 사용한 정보는 다음 페이지에서 기억하지 못하게 되며 사용자 정보 또한 지속적으로 유지할 수 없게 된다.

//추후 쿠키와 세션을 통해 요청자를 식별하게 된다.

이런 이유로 HTTP를 비 연결지향형 프로토콜이라고 부른다. 웹 브라우저는 HTTP에 맞게 웹 서버에 요청(Request)을 보내고 웹 서버는 그 요청에 따른 응답 (Response)을 HTTP에 맞게 웹 브라우저로 보내준다. 이 때 사용되는 HTTP 메소드(요청방식) 에는 Get, Post, Head, Put, Delete, Trace, Options 등 여러 가지가 있지만 웹브라우저에서 일반적으 로 사용되는 메서드는 Get과 Post 이므로 이 두 가지 메소드에 대해서만 알아볼 것이다.

- Get 메소드
  - HTTP header에 정보를 실어 보내며 클라이언트의 요청시 별도로 메소드를 지정하지 않으면 기본 - 28 - 적용 되는 메소드이다. Get 메소드는 클라이언트가 서버에 요청시 필요 한 데이터를 URL 뒤에 붙 여 전송하므로 전송되는 데이터가 사용자의 눈에 노출되어 보안성 이 좋지 않으며 요청 데이터의 크기도 8000자(Chrome, 브라우저 마다 상이함) 정도 가능하다.
- Post 메소드
  - HTTP body에 정보를 실어 보내므로 데이터 크기의 제한이 없으며 사용자의 눈에 보이지 않기 때 문에 Get 방식의 요청보다 보안성 좋다.

2.2 서블릿의 동작원리

Java Resource > Libraries > Server Runtime [Apache Tomcat v9.0] 에서 관련 파일은 확인 가능하다.

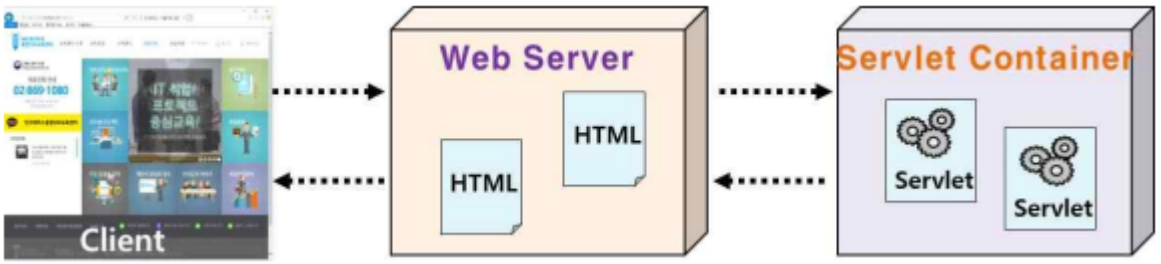
javax.servlet 패키지는 프로토콜에 독립적인 서블릿 클래스를 만들기 위해 필요한 클래스를 제공하 며 javax.servlet.http 패키지는 HTTP의 고유한 기능(GET, POST)을 제공하는 서블릿 클래스를 만 들기 위해 필요한 클래스들을 제공한다. 모든 서블릿 클래스는 javax.servlet.Servlet 인터페이스를 구현해야 한다. 이 Servlet 인터페이스를 구현한 서블릿 클래스가 javax.servlet.GenericServlet 클래스이고 GenericServlet을 상속받아 구현한 클래스가 javax.servlet.http.HttpServlet 클래스이다. HTTP 요청을 처리하기 위한 서블릿 클 래스를 작성하려면 이 HttpServlet 클래스를 상속받아 구현하면 된다. 서블릿 클래스는 웹 브라우저의 호출에 의해서 실행되기 때문에 main() 메소드가 없으며 그 대신 HttpServlet 클래스에 구현된 service() 메소드가 서블릿 컨테이너(톰캣)에 의해 호출된다. 이 service() 메소드는 특별한 경우가 아니면 자식 클래스에서 재정의(Override) 하지 않고 대신 GET 방식의 요청을 처리하기 위해 doGet() 메소드와 POST 방식의 요청을 처리하기 위해 doPost() 메소 드를 자식 클래스에서 재정의 하면 된다. 이 두 메소드는 javax.servlet.http 패키지의 HttpServletRequest 객체와 HttpServletResponse 객체를 파라미터로 넘겨받아 HTTP 요청과 응답 에 대한 처리를 구현한다.

1) 일반적인 HTML 파일 요청시 처리과정(정적 파일)

클라이언트의 요청이 웹 서버에 전달되면 해당 HTML 문서가 웹 서버에 존재하는지 검사한 후 요청 한 HTML 문서가 존재하면 웹 서버가 직접 그 HTML 문서를 클라이언트에게 응답 데이터로 전송한 다. HTML 문서가 존재하지 않으면 문서가 존재하지 않는다는 메시지가 출력된다.

2) 서블릿 요청시 처리과정(동적 파일) //ex:Calendar

클라이언트로부터 서블릿에 대한 요청이 들어오면 웹 서버는 클라이언트의 요청 정보를 묶어서 WAS 또는 서블릿 컨테이너(Servlet Container, 이하 서블릿 컨테이너)에게 처리를 의뢰한다.



Tomcat 이 WebServer 와 Servlet Container 를 포함한다.

서블릿 컨테이너는 웹 서버로부터 전달 받은 요청 정보를 분석한 후 요청에 맞는 서블릿을 검색한 다. 서블릿이 컨테이너에 존재하면 서블릿을 실행하여 클라이언트의 요청을 처리하고 그 결과를 HTML 문서로 만들어 그림같이 웹 서버를 통해 클라이언트에게 응답하게 된다.

아래의 동작은 Tomcat이 자동으로 해준다.

- 요청 및 응답 객체 생성
  - 서블릿 컨테이너는 웹 서버로부터 넘겨받은 요청 정보를 바탕으로 HttpServletRequest 객체와 HttpServletResponse 객체를 생성한다.

```
HttpServletRequest request = new HttpServletRequest();
HttpServletResponse response = new HttpServletResponse();
```

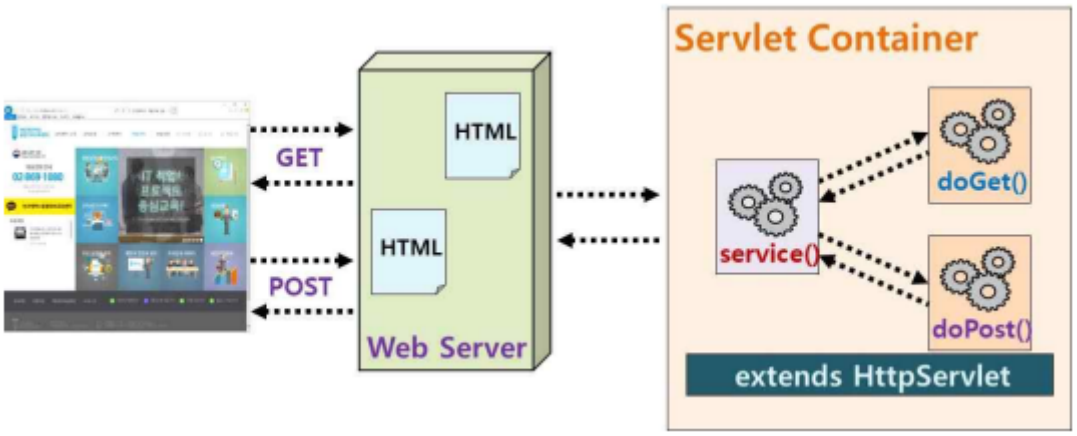
- 서블릿 객체 생성
  - 서블릿 컨테이너는 서블릿 클래스가 이미 서블릿 컨테이너에 로딩 되어 있다면 바로 해당 서블릿 클래스의 객체를 생성하고 로딩 되어 있지 않다면 해당 서블릿 클래스 파일(.class)을 로딩한 후 서 블릿 객체를 생성한다.

```
UserServlet servlet = new UserServlet();
```

- 서블릿 실행
  - 서블릿 컨테이너는 서블릿 객체를 생성한 후 그 서블릿 객체에 정의되어 있는 doGet(), doPost() 메 소드를 호출하여 사용자 요청에 대한 응답을 처리하게 된다.

```
servlet.doGet(request, response);
servlet.goPost(request, response);
```

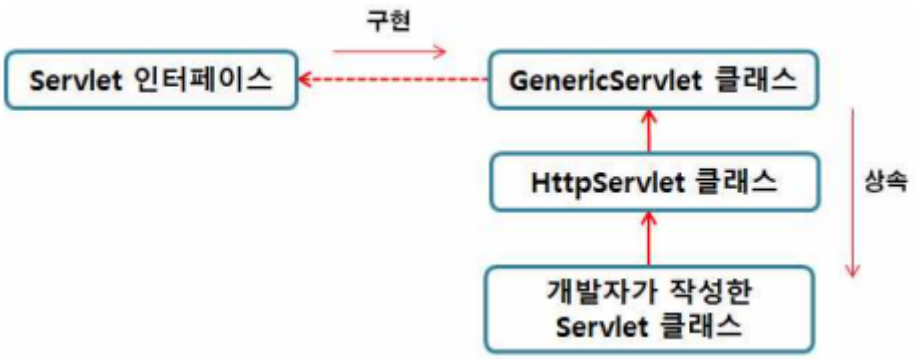
일반적으로 서블릿 클래스는 HttpServlet 클래스를 상속받아 구현하게 되는데 서블릿 컨테이너는 사용자 요청에 대한 처리를 위해서 개발자가 구현한 서블릿 클래스의 doGet() 메소드와 doPost() 메 소드를 직접 호출하는 것이 아니라 개발자가 구현한 서블릿 클래스의 슈퍼 클래스인 **HttpServlet** 클래스의 **service()** 메소드를 통해 **doGet()** 메소드와 **doPost()** 메소드를 간접적으로 호출하게 된다. 이 HttpServlet 클래스는 GenericServlet 클래스를 상속받아 HTTP 프로토콜에 적합하게 구현된 서블릿 클래스이다.



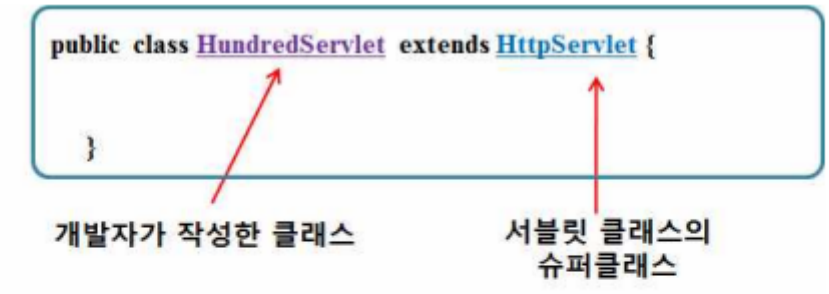
3. 서블릿 클래스 작성

3.1 서블릿 클래스 작성하기

개발자가 작성하는 서블릿 클래스의 상속 관계를 보면 아래 그림과 같으며 HTTP 서비스를 위 한 서블릿 클래스를 작성할 때는 그림 4-5와 같이 javax.servlet.http.HttpServlet 클래스를 상속하도록 만들어야 하며 public 클래스로 선언해야 한다. 또한 서블릿 클래스 안에서 GET방식 요청을 처리하기 위해서 doGet() 메소드를 재정의 하고 POST 방식 요청을 처리하기 위해서는 doPost() 메 소드를 재정의해야 한다.



그림의 doGet() 메소드와 같이 첫 번째 파라미터로 javax.servlet.http.HttpServletRequest 타 입을 받고 두 번째 파라미터로 javax.servlet.http.HttpServletResponse 타입을 받을 수 있도록 정의하고 javax.servlet.ServletException과 java.io.IoException을 선언해야 한다. 이들 Exception 처리가 필요치 않으면 생략할 수 있지만 다른 Exception은 선언할 수 없다. 사용자 요청에 대한 여러 가지 정보는 doGet() 메소드 또는 doPost() 메소드의 첫 번째 파라미터인 HttpServletRequest 객체를 통해서 구할 수 있으며 사용자 요청에 대한 응답 처리는 두 번째 파라미터인 HttpServletResponse 객체를 통해서 브라우저로 응답할 데이터를 작성하면 된다.




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

서블릿 클래스 작성

//요약 : 서블릿은 **extends** HttpServlet 클래스로 상속받는데, do(get) 과 do(post) 타입 중 하나의 파라미터로 request 는 요청을 받고, response는 요청을 처리해준다. 예외처리는 **throws** ServletException, IOException 를 통해 처리한다.

3-2) com.jspstudy.ch04.servletbasic.NowServlet

[]실습파일 생성

- 프로젝트 : 빈공간 우클릭 > New > Other… > Web > Dynamic Web Project > JSPClassCh04
- 패키지 : src/main/java > com.jspstudy.ch04.servletbasic
- 클래스 : NowServlet.java > Superclass : javax.servlet.http.HttpServlet

[]삭제시 주의사항 : Project Explorer에서의 삭제는 디스크 삭제가 아니므로, 아예 파일 없애거나 체크

클래스 파일 생성때부터 Superclass : javax.servlet.http.HttpServlet 를 하면 일부태그가 작성된다.

```
package com.jspstudy.ch04.servletbasic;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Calendar;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
//@WebServlet();
public class NowServlet extends HttpServlet {
	//get 방식 요청을 처리하는 메서드 재정의
	public void doGet(HttpServletRequest request,
			HttpServletResponse response) throws ServletException, IOException {
	//ByteStream, CharacterStream
	PrintWriter out = response.getWriter();
	Calendar cal = Calendar.getInstance();
	out.println("<h2>서블릿 작성하기</h2>");
	out.println("Hi Servlet");
	// 작업이 끝나면 스트림을 닫는다. = 자원해제
	out.close();
	}}
```

3-2)* 서블릿 클래스 등록하기 //어노테이션방식이 간결해서 잘안쓰임

서블릿 클래스가 실행되기 위해서는 JSP 페이지와 달리 등록 과정이 필요하다. 서블릿 클래스는 웹 애플리케이션의 배포 서술자(deployment descriptor) 파일에 등록해야 한다. 배포 서술자 파일이란 웹 애플리케이션 디렉터리 아래 WEB-INF 디렉터리에 위치하는 web.xml 파일을 말하며 이 web.xml 파일은 하나의 웹 애플리케이션에 하나만 만들 수 있다.

web.xml > Source 에서 아래를 작성

```
 </welcome-file-list>
<servlet>
 <servlet-name>nowServlet</servlet-name>
 <servlet-class>com.jspstudy.ch04.basic.NowServlet</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>now-servlet</servlet-name>
<url-pattern>/now.do</url-pattern>
</servlet-mapping>
</web-app>
```

서블릿 3.0부터는 web.xml 파일 외에 어노테이션(Annotation)을 이용한 URL 맵핑 기술을 지원 한다.

3-3) com.jspstudy.ch04.servletbasic.AnnotationNowServlet - 어노테이션을 이용한 서블릿 등록하기

어노테이션은 자바 5.0부터 지원하는 기술로 서블릿 3.0부터 web.xml에 설정해야 하는 내용들을 web.xml에 설정하지 않고 소스 코드에서 어노테이션을 사용해 설정할 수 있도록 지원하고 있다. 앞에서 작성한 NowServlet 클래스를 복사하여 AnnotationNowServlet으로 이름을 지정하고 다음과 같이 클래스 선언부 위에 @WebServlet 어노테이션을 이용해 서블릿 매핑을 설정하고 설정한 서블릿 매핑에 맞게 실행되는지 테스트 해보자.

```
package com.jspstudy.ch04.servletbasic;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Calendar;
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({"/time", "/now"})
public class AnnotationNowServlet extends HttpServlet {

 //GET 방식 요청을 처리하는 메서드 재정의 (오버라이드)
 public void doGet(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException{

 // 웹 브라우저에 출력될 응답문서의 형식과 문자 셋을 지정 * ContentType 설정은 Response 객체로
 부터 PrintWriter * 객체를 얻기 전에 지정해야 한글이 깨지지 않고 제대로 동작한다.
   response.setContentType("text/html; charset=utf-8");

   //ByteStream, CaracterStream, 요청에 대한 처리 결과를 웹 브라우저에 출력할 스트림객체 얻기
   PrintWriter out = response.getWriter();
   Calendar ca = Calendar.getInstance();

   out.println("<h2>어노테이션 서블릿 작성하기</h2>");
   out.println("Hi Servlet");

   //작업이 끝나면 스트림을 닫아줘야함 = 자원을 해제한다.
   out.close();
 }}
```

웹 애플리케이션 이름이 “JSPStudyCh04”이므로 로컬 네트워크에서 접속한다면 브라우저 주소창에 “http://localhost:8080/JSPStudyCh04/now”를 입력하면 실행된다. 예제는 클라이언트의 요청을 서블릿에서 처리하고 응답 데이터를 저장할 수 있는 Response 객체를 통해서 클라이언트로 출력해 주는 스트림 객체인 PrintWriter 객체를 구해서 서블릿 클래스에서 직접 웹 브라우저로 전송할 데이터를 HTML 문서 형식에 맞게 작성하였다. 응답 결과가 단순 하기 때문에 서블릿 클래스에서 HTML 문서 형식으로 작성하는 것이 그렇게 복잡하지 않다고 생각 할 수 있겠지만 여러 가지 문서 객체를 포함하는 복잡한 HTML 문서를 서블릿 클래스에서 직접 작 성하는 것은 JSP 페이지에서 작성하는 것에 비해서 매우 비효율적이라 할 수 있다. 그러므로 서블릿 클래스에서 요청을 처리하고 그 결과인 응답 데이터를 **JSP** 페이지로 보내서 웹 브라우저 화면에 보여지는 부분을 JSP에서 처리하면 서블릿 클래스는 위에서 보다 훨씬 간결해 질 것이다.

3-4) URL 패턴 맵핑 com.jspstudy.ch04.urlmapping - package 참고 (생략)
@WebServlet("/mapping2/*") 을 통해 해당 요청은 해당하는 곳에서 모두 처리하는 유형

4. 요청방식(GET, POST)에 따른 처리

4-1. GET방식 요청 처리

패키지 : com.jspstudy.ch04.servletbasic
서블릿파일 : getRequestServlet

```
package com.jspstudy.ch04.servletbasic;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Calendar;
import javax.servlet.RequestDispatcher;
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({"/time", "/now"})
public class AnnotationNowServlet extends HttpServlet {

 //GET 방식 요청을 처리하는 메서드 재정의 (오버라이드)
 public void doGet(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException{

            //사용자가 보낸온 데이터 - 요청 파라미터
            String name = request.getParameter("name");

            //요청을 받아서 처리하는 일 - 다른 클래스를 사용
            //최종적으로 나온 결과 - 화면 출력해야 되는 데이터 - 모델
            request.setAttribute("title", "서블릿으로 받은 제목" );
            request.setAttribute("message", "안녕JSP");
            request.setAttribute("name", name);

            //제어를 view 페이지(JSP)로 이동시켜서 출력
            //서버에서 서버의 또 다른 모듈 - forward
            RequestDispatcher rd = request.getRequestDispatcher("/view/view.jsp");
            rd.forward(request, response);
 }}
```

4-1*. GET방식 요청 처리

1) JSPStudyCh04/src/main/webapp/getRequestServletForm.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>GET 방식 요청처리</title>
</head>
<body>
```

```
<!-- form 태그의 method 속성을 생략했기 때문에 default인 get 방식이 적용된다. 브라우저는 현재 문서의 문자 셋을 기준으로 폼 데이터를 인코딩하여 서버로 전송한다.-->
        <form name="f1" action="getRequest" >
                첫 번째 숫자 : <input type="number" name="num1" min="1" /><br/>
                두 번째 숫자 : <input type="number" name="num2" min="1" /><br/>
                <input type="reset" value="다시쓰기" />
                <input type="submit" value="전송하기" />
        </form></body></html>
```

- 액션은 어디로 보낼지를 결정, 메소드 포스트는 디폴트라 생략가능
- action="getRequest" method="Post"

2) JSPStudyCh04/src/main/webapp/getRequestServlet.java
패키지 : com.jspstudy.ch04.servletbasic
서블릿파일 : getRequestServlet

```
package com.jspstudy.ch04.requestmethod;
import java.io.IOException;
import java.io.PrintWriter;
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("/getRequest")
public class getRequestServlet extends HttpServlet {
        private static final long serialVersionUID = 1L;
        protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

                //요청 처리
                String num1 = request.getParameter("num1");
                String num2 = request.getParameter("num2");
                int firstNum = Integer.parseInt(num1);
                int secondNum = Integer.parseInt(num2);

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

                //응답데이터를 쓰기위한 스트림 객체 구하기
                PrintWriter out = response.getWriter();

                //html 포멧에 맞춰서 응답 데이터 작성
                out.println("<h2>get 방식 요청 처리</h2>");
                out.println("<P>첫 번째 입력값 : " + firstNum + "</p>");
                out.println("<P>두 번째 입력값 : " + secondNum + "</p>");
                out.println("<P>두 수를 곱한값 : " + firstNum * secondNum + "</p>");
                out.close();
        }}

//get 방식은 header 로 본문이 없이 가는 것
```

4-2. POST방식 요청 처리

POST방식의 요청은 HTTP body에 데이터를 실어 보내는 방식으로 데이터 크기의 제한이 없고 사용자의 눈에 보이지 않아 GET방식 보다 보안성이 뛰어나다. GET방식에서는 데이터를 URL 뒤에 추 가하여 전송하였으나 POST 방식은 HTTP 요청 본문에 데이터를 포함하여 전송한다. 즉 HTTP 요 청 본문에 첨부파일 형태로 데이터를 추가하여 전송하기 때문에 데이터 크기에 제한이 없다

1) JSPStudyCh04/src/main/webapp/postRequestServletForm.jsp

2) com.jspstudy.ch04.requestmethod.PostRequestServlet.java