

## 3. Servlet

1. Servlet 개요
2. Servlet 프로그램 작성
3. Servlet에서 한글 처리
4. 다양한 입력양식 값처리

# 1. Servlet 개요

## □ Servlet 개요

- ▣ 브라우저는 기본적으로 HTML 형식의 문서만을 표시, 동적인 처리 불가!
- ▣ 웹 서버 측에서 사용자의 요구에 따라 자동으로 생성된 HTML 형식의 페이지를 생산해 전송해 줄 수 있는 여러 기술이 개발됨
- ▣ 자바 진영의 기술이 서블릿(Servlet)
- ▣ 서블릿: 웹 서버 상에서 실행되는 자바의 클래스 파일, 기본적으로 자바의 모든 API를 그대로 사용할 수 있으며 강력한 객체지향성 등 자바의 장점을 모두 가질 수 있음

# 1. Servlet 개요

## □ Servlet 자바 클래스와 차이점

- ▣ 서블릿은 반드시 javax.servlet. Servlet 인터페이스를 구현 (Implements)해서 작성해야만 함
- ▣ 입력과 출력을 HTTP 프로토콜의 요청(Request)과 응답 (Response)의 형태로 다룸
- ▣ 서블릿은 서버사이드의 자바 응용 프로그램!

# 1. Servlet 개요

## □ HTTP 프로토콜

- ▣ 인터넷 통신 프로토콜 : TCP/IP, FTP, SMTP, HTTP 등
- ▣ HTTP(HyperText Transfer Protocol)은 가장 대표적인 웹브라우저 통신에 관한 프로토콜
- ▣ HTTP 요청과 응답 구조

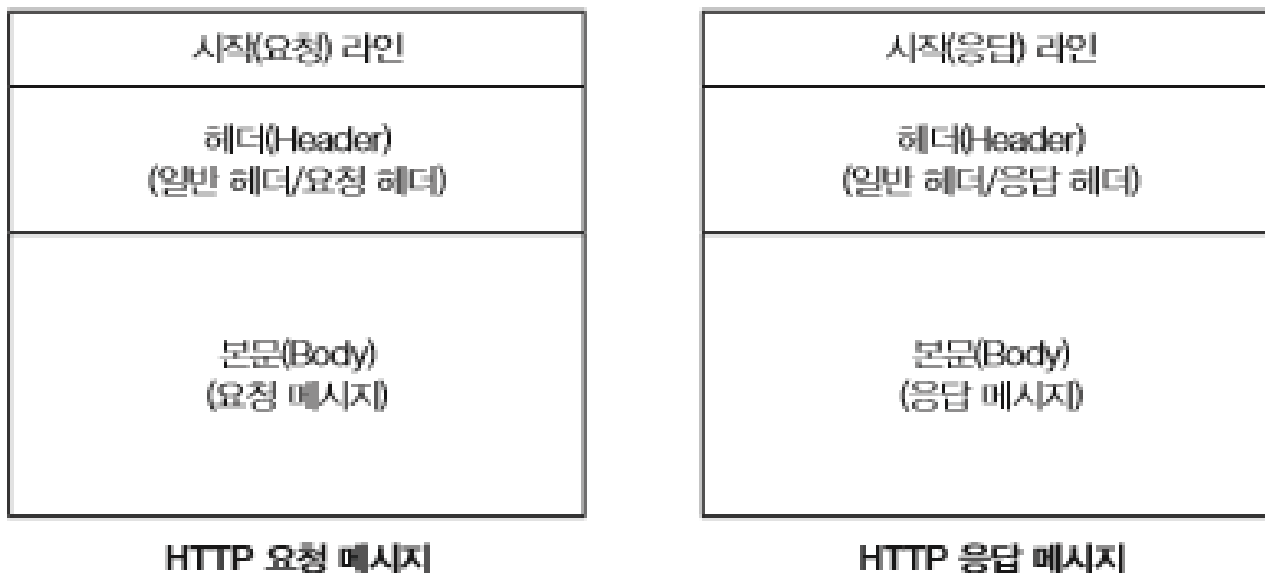


그림 1-6. HTTP 메시지의 구조

# 1. JSP와 Servlet이란?

## □ HTTP 요청(Request) 메시지

- HTTP 요청 메시지: HTTP 메소드(Method), 접근할 주소(URL) 정보, 서버에 전달할 데이터인 폼 파라미터로 구성
- 대표적인 HTTP 메소드 : GET 메소드, POST 메소드
  - Get 메소드 사용 → Get 방식 요청
  - Post 메소드 사용 → Post 방식 요청
- **Get 방식 요청 :**
  - 전송할 파라미터 값들을 시작 라인의 URL 정보에 붙여서 같이 전송 → 데이터 크기 제약, 속도 빠름, 보안 취약
- **Post 방식 요청 :**
  - 파라미터 값들을 요청 메시지의 본문(Body)에 담아서 전송 → 데이터 크기 제약 無, 속도 느림, 보안 강함

# 1. Servlet 개요

## □ HTTP 요청(Request) 메시지

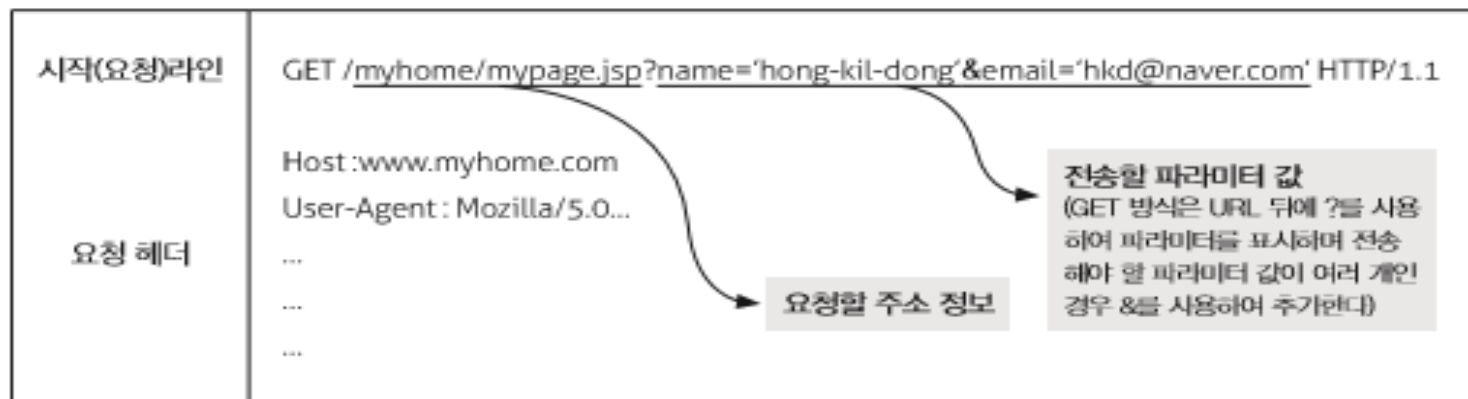


그림 1-7. GET 방식의 요청 메시지

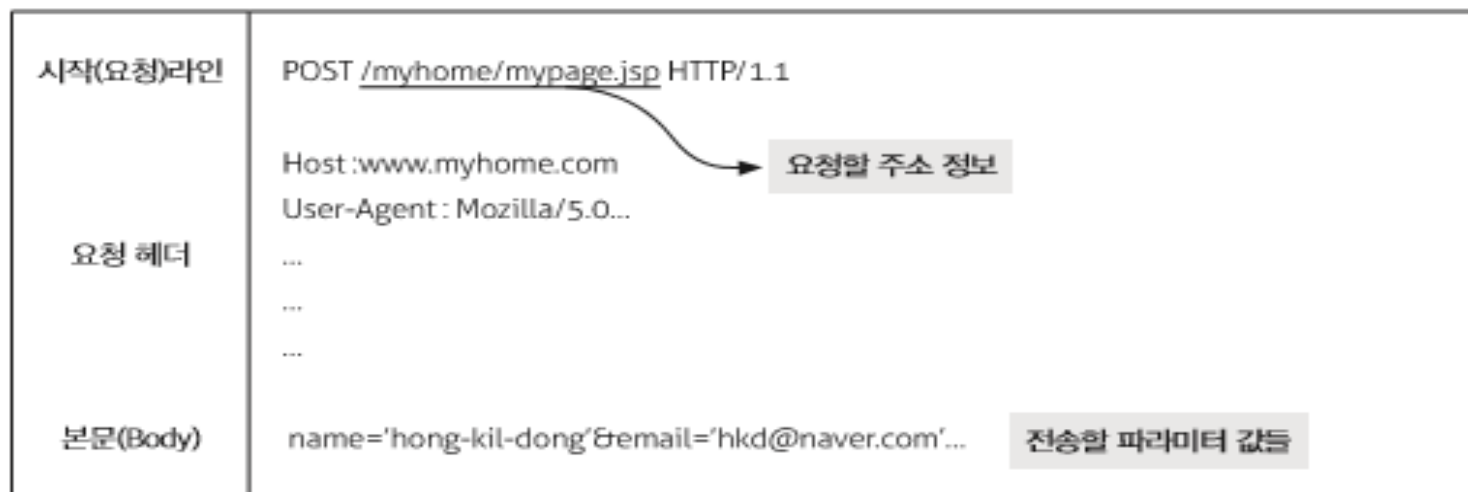


그림 1-8. POST 방식의 요청 메시지

# 1. Servlet 개요

## □ HTTP 응답(Response) 메시지

- HTTP 응답 메시지 : 요청에 대한 서버의 처리 성공 여부를 표시하는 상태 코드(HTTP 404, 500 등) 번호와 웹 서버가 응답해주는 콘텐츠의 타입 정보(텍스트/HTML, 이미지 등), 콘텐츠의 내용으로 구성
- 서블릿 클래스가 요청을 처리해 생성하는 페이지는 웹 서버에서 응답 메시지의 형태로 작성되어 사용자의 브라우저에 전송

# 1. Servlet 개요

## □ 웹 컨테이너

- ▣ JSP 파일의 실행 요청을 처리
- ▣ 웹 서버 내부에서 서블릿 클래스 또는 JSP 파일을 실행하기 위한 실행 환경을 제공하는 역할
- ▣ 어떤 주소에(URL) 대한 요청 → HTTP 서버 → 그 주소에 미리 매핑되어 있는 콘텐츠(HTML 파일이나 이미지 등)를 사용자의 브라우저에 응답 형태로 전송
- ▣ 요청된 URL이 서블릿 클래스 또는 JSP 파일일 경우 HTTP 서버 → 웹 컨테이너



# 1. Servlet 개요

## □ 서블릿(Servlet) 동작 원리

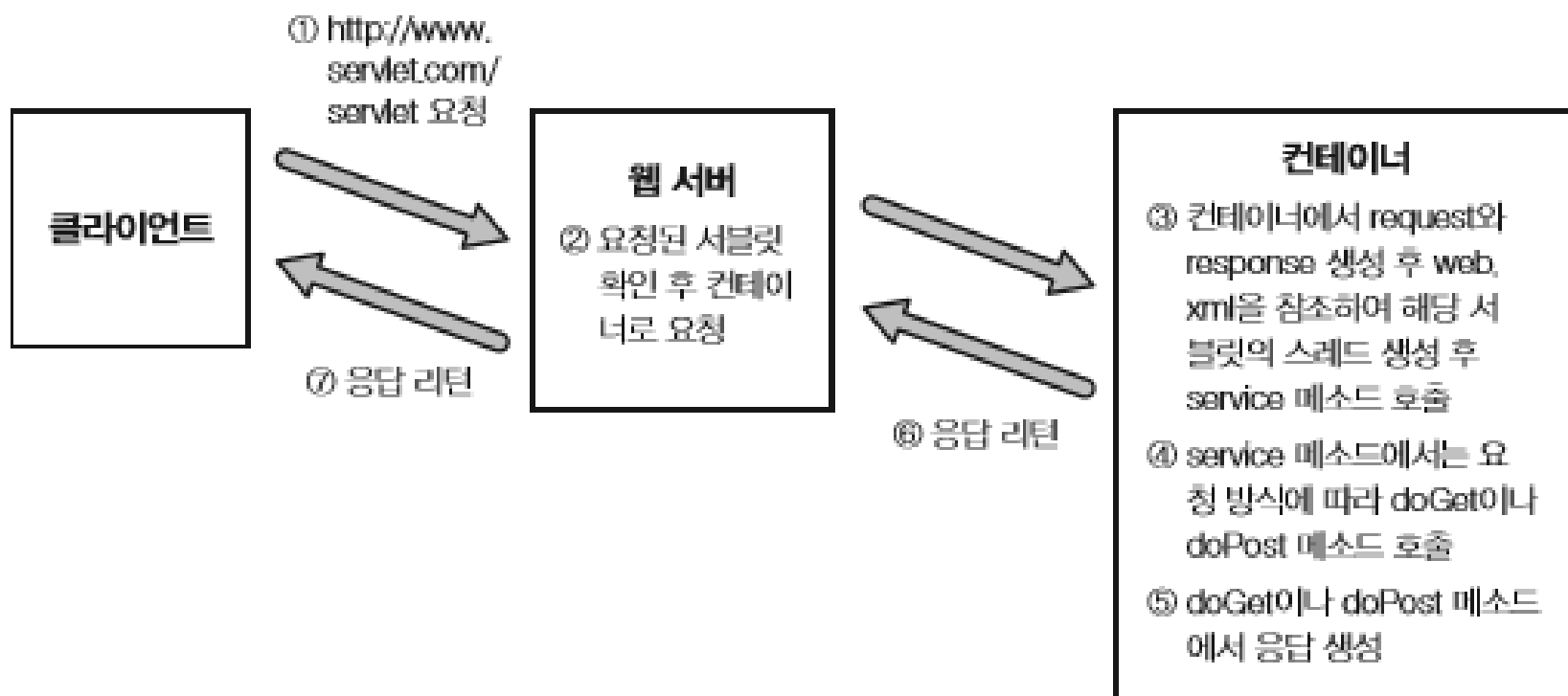


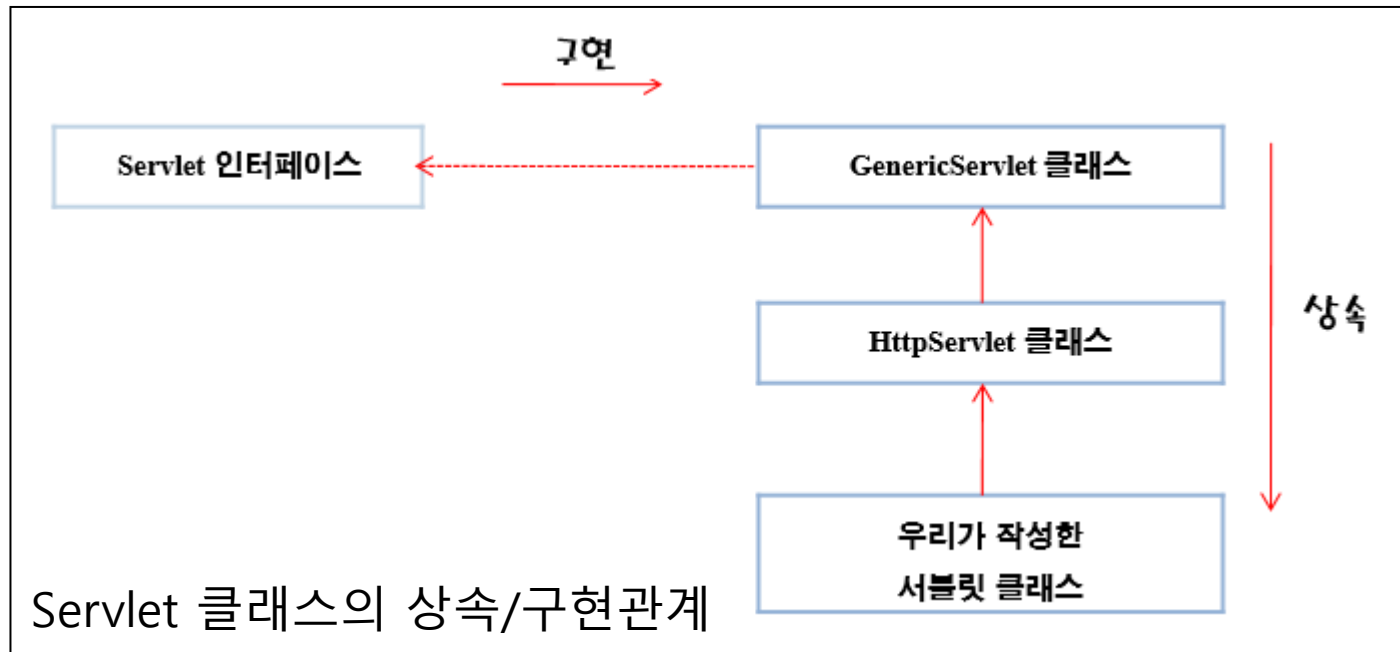
그림 1-6. Servlet 요청 처리 단계

## 2. 서블릿 클래스의 작성

### □ 서블릿 클래스의 작성을 위한 준비

#### ▣ 서블릿 클래스를 작성할 때 지켜야 할 규칙 세 가지

- 서블릿 클래스는 javax.servlet.http.HttpServlet 클래스를 상속하여 작성
- doGet 또는 doPost 메서드 안에 웹 브라우저로부터 요청이 왔을 때 해야 할 일을 기술
- HTML 문서는 doGet, doPost 메서드의 두 번째 파라미터를 이용해서 출력



# Servlet 클래스 작성 예

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

    public HelloServlet() {
        super();
    }

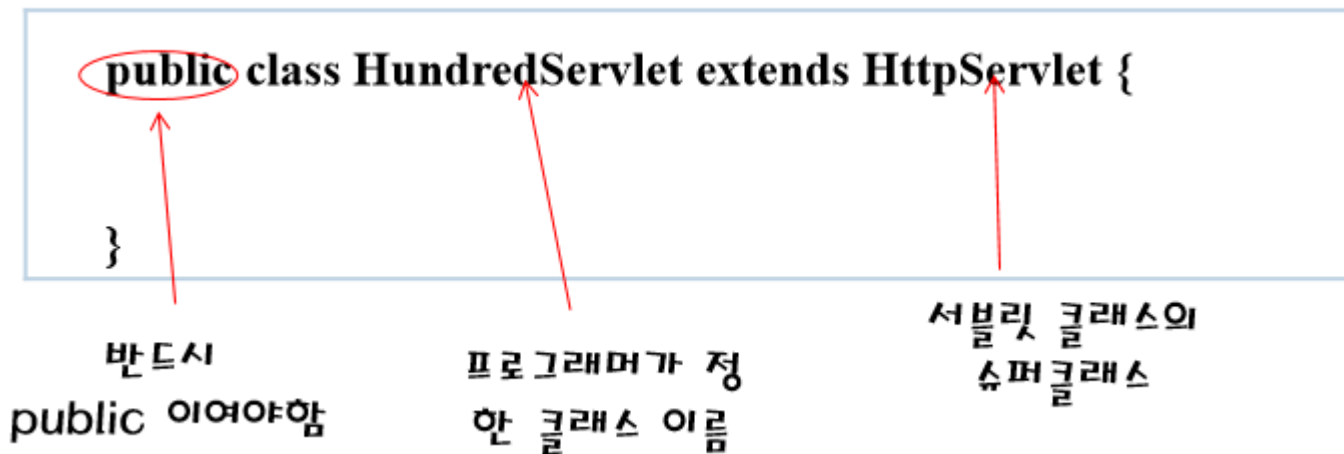
    // @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        // TODO Auto-generated method stub
        response.getWriter().append("Served at: ").append(request.getContextPath());
    }

    // @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        // TODO Auto-generated method stub
        doGet(request, response);
    }
}
```

## 2. 서블릿 클래스의 작성

### □ 서블릿 클래스 정의

- ▣ javax.servlet.http.HttpServlet 클래스를 상속받도록 작성
- ▣ 반드시 public 클래스로 작성.



## 2. 서블릿 클래스의 작성


### □ Servlet 클래스

- ▣ 서블릿 클래스 안에 doGet 또는 doPost 메서드를 선언해야 함
- ▣ 두개의 파라미터 javax.servlet.http.HttpServletRequest 와 javax.servlet.http.HttpServletResponse 받음
- ▣ 예외 처리를 위해 javax.servlet.ServletException, java.io.IOException을 던짐
  - javax.servlet.http.HttpServletRequest : 클라이언트의 요청 처리
  - javax.servlet.http.HttpServletResponse : 요청처리 결과를 클라이언트에게 응답 처리

## 2. 서블릿 클래스의 작성

- 서블릿 클래스 정의
  - ▣ doGet() 메소드

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



HttpServlet 클래스의 doGet 메서드와  
리턴 타입, 파라미터 변수, 익셉션 타입이 동일해야 한다.

- doGet 메서드를 public으로 선언 : 웹 컨테이너가 웹 브라우저로부터 요청을 받아서 메서드를 호출할 때 필요.
- doGet 메서드의 throws 절의 ServletException, IOException은 생략 가능, 다른 익셉션을 추가할 수는 없다

## 2. 서블릿 클래스의 작성

- 결과 출력 HTML 문서의 종류와 인코딩 방식 지정

```
response.setContentType("text/html;charset=euc-kr");
```

- 출력 스트림 얻기

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

- 클라이언트 결과 출력

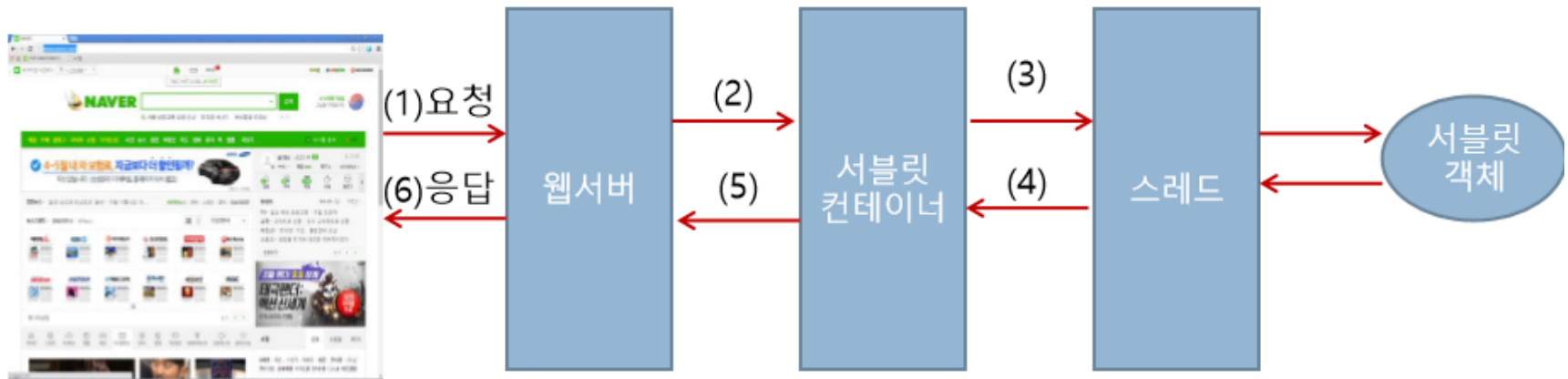
```
out.println(page + " 페이지 게시판 목록 출력");
```

- 출력 스트림 닫기

```
out.close();
```

## 2. 서블릿 클래스의 작성

### □ 서블릿 동작원리



- ① 브라우저가 서브릿 요청
- ② WAS안에 웹 서버가 서블릿 요청을 인식하여 서블릿 컨테이너에게 서브릿을 수행하도록 넘겨줌
- ③ 서블릿은 스레드를 기동하여 해당 서브릿 객체를 생성하여 이를 수행
- ④ 서블릿 객체의 작업이 종료되면 기동되었던 스레드가 종료
- ⑤ 서블릿 수행 결과가 웹 서버에 전송됨
- ⑥ 이를 클라이언트에게 전송



## 2. 서블릿 클래스의 작성

### □ 쿼리 스트링

- ▣ 사용자가 입력한 데이터를 서버로 전달하는 방법
- ▣ get 방식, post 방식
- ▣ get 방식
  - URL 주소뒤에 입력 데이터를 함께 보냄

리소스 ? 변수=값

ParamServlet?id=pinksung & age=15

- 요청 객체(request)와 파라미터 관련 메소드(getParameter

```
String id=request.getParameter("id");
```

```
Int age=Integer.parseInt(request.getParameter("age");
```

# 예제 작성

- 예제1 : 두 수를 더한 결과 출력 Servlet 작성
  - ▣ doGet 방식으로 작성
  - ▣ doPost 방식으로 작성
  
- 예제2: 클래스명 : HundredServlet
  - ▣ 1-100까지 더하여 결과를 출력을 doGet() 메소드에 작성하라.
  
- 예제3: 클래스명 : LoginServlet
  - ▣ 아이디와 패스워드 전송 폼을 작성하고 Servlet에서 전송된 아이디와 패스워드를 출력(doGet과 doPost 방식으로 작성)

## □ 서블릿 라이프 사이클



# 서블릿의 라이프 사이클 테스트

- 클래스 명: Lifecycle, URL Mapping: 기본
- Which method stubs would you like to create? Init(), destroy()를 추가 체크한다.

```
@WebServlet("/LifeCycle")
public class Lifecycle extends HttpServlet {
    private static final long serialVersionUID = 1L;
    int initCount=1;
    int doGetCount=1;
    int destroyCount=1;

    public Lifecycle() {
        super();
    }
    public void init(ServletConfig config) throws ServletException {
        System.out.println("init 메소드는 첫 요청만 호출됨:" + initCount++);
    }
    public void destroy() {
        System.out.println("destroy 메소드는 콧이 종료될때만 호출됨" + destroyCount++);
    }
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        System.out.println("doGet 메소드가 요청할때마다 호출됨:" + doGetCount++);
    }
}
```

# 3. Servlet의 한글처리

## □ 응답시 한글 처리

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

## □ 요청 데이터 한글처리

### ▣ get 방식 :

- GET 방식인 경우는 페이지 에서 사용하고 있는 캐릭터셋으로 인코딩되어 파라미터가 전송됨
- 톰캣의 URL 인코딩 방식은 UTF-8 방식 → GET 방식은 클라이언트의 페이지 인코딩 방식이 UTF-8 이어야 함
- 이클립스 Preference 메뉴에서 UTF-8 로 설정

### ▣ post 방식

- POST 방식인 요청 파라미터 값이 요청 body 영역에 따로 인코딩되어 넘어오므로 URLEncoding 로 처리 불가
- request 객체의 setCharacterEncoding 메소드를 사용하여 request 객체의 body 영역의 인코딩 방식을 변경

```
request.setCharacterEncoding("UTF-8");
```

# 예제 작성

- 예제1 : 회원가입 폼을 작성하여 회원정보를 받아서 출력하는 Servlet를 작성한다.
  - ▣ memregForm.html
  - ▣ MemregServlet
  - ▣ 회원 정보 : 이름, 나이 주소, 전화번호
  - ▣ doGet, doPost 두가지 방식으로 작성

## 4. 다양한 입력양식 처리

### □ 유효성 체크

#### ▣ Param01.jsp

```
<form method="get" action="ParamServlet" name="frm">  
아이디 : <input type="text" name="id"> <br>  
나 &nbsp; 이 : <input type="text" name="age"> <br>  
<input type="submit" value="전송" onclick="return check()">  
</form>
```

## 4. 다양한 입력양식 처리

### □ 유효성 체크

#### ▣ ParmServlet

```
protected void doGet(HttpServletRequest request,
    HttpServletResponse response) throws ServletException, IOException {
    response.setContentType("text/html;charset=utf-8");
    String id = request.getParameter("id");
    int age = Integer.parseInt(request.getParameter("age"));
    PrintWriter out = response.getWriter();
    out.print("<html> <body>");
    out.println("당신이 입력한 정보입니다.<br>");
    out.println("아 이 디 : ");
    out.println(id);
    out.println("<br> 나이 : ");
    out.println(age);
    //자바스크립트로 이전 페이지로 이동하는 링크를 만들어 줌
    out.println("<br> <a href='javascript:history.go(-1)'>다시 </a>");
    out.print("</body> </html>");
    out.close();
}
```



## 4. 다양한 입력양식 처리

### □ 유효성체크

#### ▣ 자바 스크립트 파일 (param.js)

```
function check() {  
    if (document.frm.id.value == "") {  
        alert("아이디를 입력해주세요.");  
        document.frm.id.focus();  
        return false;  
    } else if (document.frm.age.value == "") {  
        alert("나이를 입력해주세요.");  
        document.frm.age.focus();  
        return false;  
    } else if (isNaN(document.frm.age.value)) {  
        alert("숫자로 입력해주세요.");  
        document.frm.age.focus();  
        return false;  
    } else {  
        return true;  
    }  
}
```

HTML에 외부 자바 스크립트 불러오기

```
<script type="text/javascript"  
src="param.js"> </script>
```

## 4. 다양한 입력양식 처리

### □ 암호 입력상자 예

```
<form method="get" action="LoginServlet">  
  <label for="userid"> 아이디 : </label>  
  <input type="text" name="id" id="userid"> <br>  
  <label for="userpwd"> 암호 : </label>  
  <input type="password" name="pwd" id="userpwd"> <br>  
  <input type="submit" value="로그인">  
</form>
```

## □ LoginServlet

```
protected void doGet(HttpServletRequest request,
    HttpServletResponse response) throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    String id = request.getParameter("id");
    String pwd = request.getParameter("pwd");
    PrintWriter out = response.getWriter();
    out.print("<html> <body>");
    out.println("당신이 입력한 정보입니다.<br>");
    out.println("아이디 : ");
    out.println(id);
    out.println("<br> 비밀번호 : ");
    out.println(pwd);
    out.println("<br> <a href='javascript:history.go(-1)'> 다시 </a>");
    out.print("</body> </html>");
    out.close();
}
```

## □ TestArea와 Radio

```
<form method="get" action="RadioServlet">
  <label for="gender"> 성별 : </label>
  <input type="radio" id="gender" name="gender" value="남자" checked>남자
  <input type="radio" id="gender" name="gender" value="여자"> 여자 <br><br>
  <label for="chk_mail"> 메일 정보 수신 여부 : </label>
  <input type="radio" id="chk_mail" name="chk_mail" value="yes" checked>수신
  <input type="radio" id="chk_mail" name="chk_mail" value="no">거부<br><br>
  <label for="content"> 간단한 가입 인사를 적어주세요^o^ </label>
  <textarea id="content" name="content" rows="3" cols="35"> </textarea> <br>
  <input type="submit" value="전송">
</form>
```

## □ RadioServlet

```
protected void doGet(HttpServletRequest request,
    HttpServletResponse response) throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    String gender = request.getParameter("gender");
    String chk_mail = request.getParameter("chk_mail");
    String content = request.getParameter("content");
    PrintWriter out = response.getWriter();
    out.print("<html> <body>");
    out.println("당신이 입력한 정보입니다.<hr>");
    out.println("성별 : <b>");
    out.println(gender);
    out.println("</b> <br> 메일 정보 수신 여부 : <b>");
    out.println(chk_mail);
    out.println("</b> <br> 가입 인사 : <b> <pre>");
    out.println(content);
    out.println("</b> </pre> <a href='javascript:history.go(-1)'>다시 </a>");
    out.print("</body> </html>");
    out.close();
}
```

## 4. 다양한 입력양식 처리

- 하나의 파라미터 이름으로 여러 개의 파라미터 값 전송 될 때 처리
  - ▣ HttpServletRequest 인터페이스에서 제공되는 `String[ ]` `getParameterValues(String paramName)` 메소드 사용
  - ▣ √ 체크 박스를 이용한 여러 개의 파라미터 전송 예제

## □ Check.jsp

```
<h2>악세사리 </h2>
  관심항목을 선택하세요.
  <hr>
  <form method="get" action="CheckboxServlet">
    <input type="checkbox" name="item" value="신발"> 신발
    <input type="checkbox" name="item" value="가방"> 가방
    <input type="checkbox" name="item" value="벨트"> 벨트 <br>
    <input type="checkbox" name="item" value="모자"> 모자
    <input type="checkbox" name="item" value="시계"> 시계
    <input type="checkbox" name="item" value="주얼리"> 주얼리 <br>
    <input type="submit" value="전송">
  </form>
```

## □ CheckServlet

```
protected void doGet(HttpServletRequest request,
    HttpServletResponse response) throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    out.print("<html> <body>");
    String items[] = request.getParameterValues("item");
    if (items == null) {
        out.print("선택한 항목이 없습니다.");
    } else {
        out.println("당신이 선택한 항목입니다.<hr>");
        for (String item : items) {
            out.print(item + " ");
        }
    }
    out.println("<br> <a href='javascript:history.go(-1)'> 다시 </a>");
    out.print("</body> </html>");
    out.close();
}
```



## □ Select.jsp

```
<form method="get" action="SelectServlet">
  <span style="float: left; margin-right: 20px"> <label for="job">직업 </label>
    <select id="job" name="job" size="1">
      <option value="">선택하세요 </option>
      <option value="학생">학생 </option>
      <option value="컴퓨터/인터넷">컴퓨터/인터넷 </option>
      <option value="언론">언론 </option>
      <option value="공무원">공무원 </option>
      <option value="군인">군인 </option>
      <option value="서비스업">서비스업 </option>
      <option value="교육">교육 </option>
    </select>
  </span> <label for="interest" style="float: left;">관심분야</label>
    <select id="interest" name="interest" size='5' multiple="multiple">
      <option value="에스프레소">에스프레소 </option>
      <option value="로스팅">로스팅 </option>
      <option value="생두">생두 </option>
      <option value="원두">원두 </option>
      <option value="핸드드립">핸드드립 </option>
    </select> <br> <br>
  <input type="submit" value="전송" style="float: right; margin-right: 50px">
</form>
```

## □ SelectServlet

```
protected void doGet(HttpServletRequest request,
    HttpServletResponse response) throws ServletException, IOException {
    String job = request.getParameter("job");
    String interests[] = request.getParameterValues("interest");
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    out.print("<html> <body>");
    out.println("당신이 선택한 직업 : <b>");
    out.print(job);
    out.println("</b> <hr>당신이 선택한 관심 분야 : <b>");
    if (interests == null) {
        out.print("선택한 항목이 없습니다.");
    } else {
        for (String interest : interests) {
            out.print(interest + " ");
        }
    }
    out.println("</b> <br> <a href='javascript:history.go(-1)'>다시 </a>");
    out.print("</body> </html>");
    out.close();
}
```

# 예제 :

회원가입 정보

아이디:	<input type="text"/>
비밀번호:	<input type="password"/>
이름:	<input type="text"/>
주소:	<input type="text"/>
전화번호:	<input type="text"/>
성별:	<input type="radio"/> 남자 <input type="radio"/> 여자
취미:	<input type="checkbox"/> 게임 <input type="checkbox"/> 등산 <input type="checkbox"/> 사이클 <input type="checkbox"/> 낚시
직업:	<input type="text" value="회사원"/>

http://localhost:8081/

파일(F) 편집(E) 보기(V) 즐겨찾기(A) >>

아이디:abcd  
비밀번호:1234  
이름:홍길동  
주소:부산시  
전화번호:010-2525-4545  
성별:F  
취미:게임등산  
직업:회사원