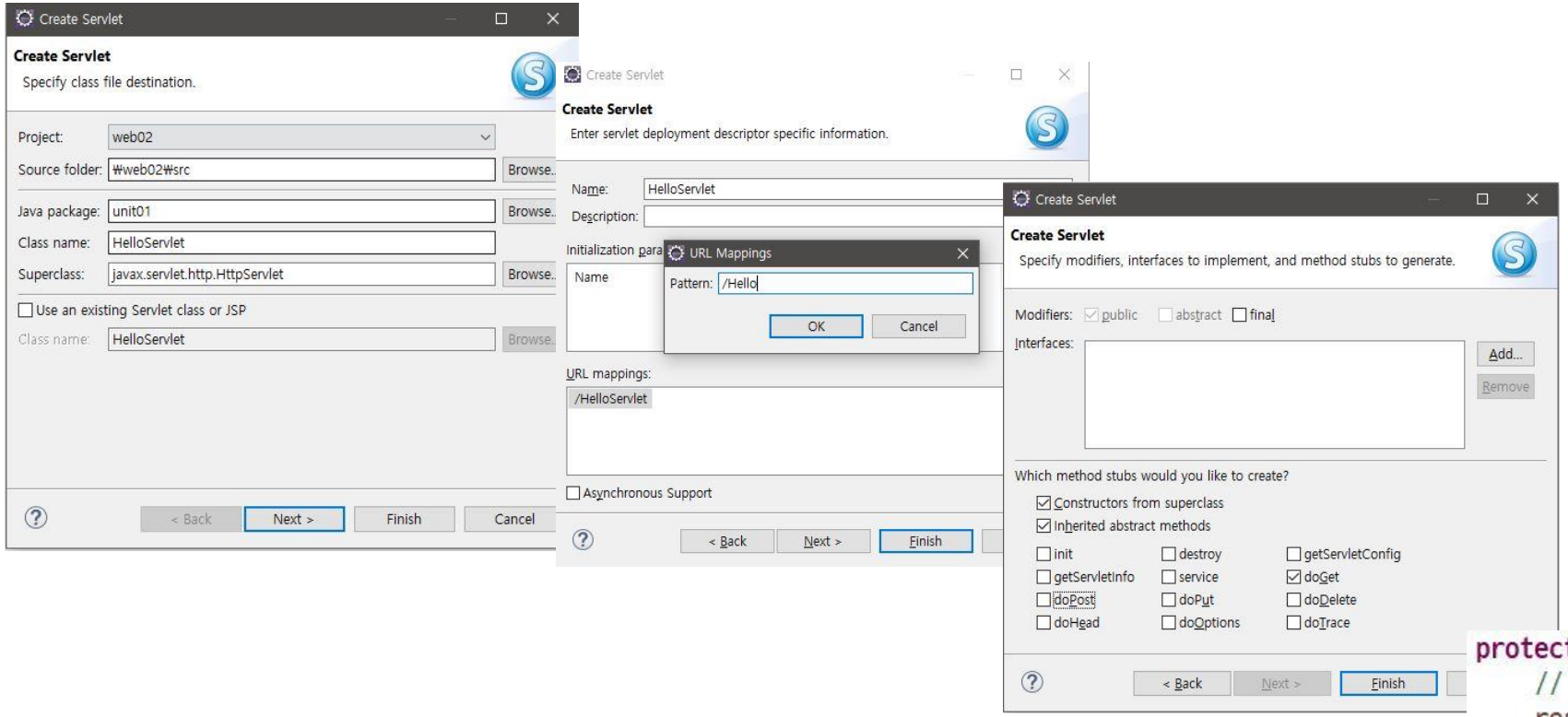
02강 서블릿 기초

서블릿 프로그램을 만들어 보자

- 간단한 서블릿 프로그램을 만들어 본다.




```
protected void doGet(HttpServletRequest request, HttpS
    // 클라이언트에서 응답할 페이지 정보를 셋팅한다.
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    out.print("<html><body><h1>");
    out.print("Hello Servlet!!");
    out.print("</h1></body></html>");
    out.close();
```

서블릿 프로그램을 만들어 보자

- 서블릿로 만들어지는 페이지를 호출하기 위해서는

http://localhost:8090/web02/Hello

로 접속하면 된다.

- 웹서버는 다양한 서비스를 제공할 수 있다
- 이런 서비스는 각각의 웹 애플리케이션으로 작성되어야 하며
- 웹 애플리케이션당 하나의 프로젝트를 생성한다.

- http://localhost:8090/ 접속후

어떤 서비스를 받을지 기술하는 내용에 따라 다르다.

- 해당 웹 애플리케이션 경로를 '컨텍스트 패스'라고 한다.
  - 컨텍스트 패스는 달리 말하면 웹서버에서 제공하는 웹 애플리케이션을 구분하기 위해 사용하는 것
  - 톰캣에 등록해주어야 한다.(server.xml)

서블릿 프로그램을 만들어 보자

- 서블릿 코드를 분석 해보자

@WebServlet("/Hello")

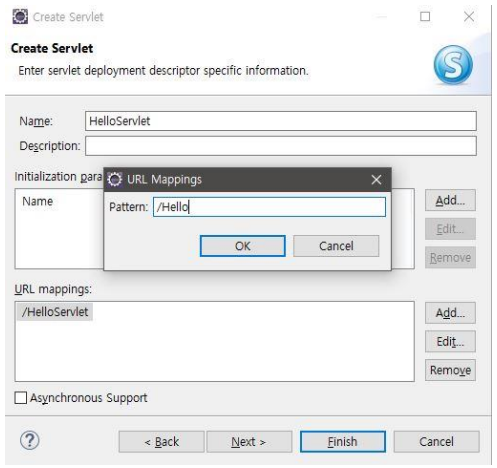
⇒서블릿 클래스 요청을 위한 URL 매핑을 보다 쉽게 처리하기 위한 어노테이션

서블릿 버전 3.0이전에는 web.xml을 직접 작성해야 한다.

http://localhost:8090/컨텍스트 패스/서블릿 매핑주소

형태로 사용된다.

매핑 주소는 기본값을 클래스 이름이지만 변경이 가능하다.



서블릿 프로그램을 만들어 보자

- 서블릿 클래스 정의하기

반드시 public으로 선언되야어야 하고 HttpServlet을 상속받아야 한다.

- HttpServlet클래스를 상속받으면 doGet과 doPost를 구현할 수 있게 된다.

- HttpServletRequest request는 요청 처리를 위한 객체
- HttpServletResponse response 응답 처리를 위한 객체이다.

- doGet, doPost메소드를 모두 구현해야 두가지 방식에 모두 대응할 수있다.

서블릿 프로그램을 만들어 보자

- 응답객체 – response

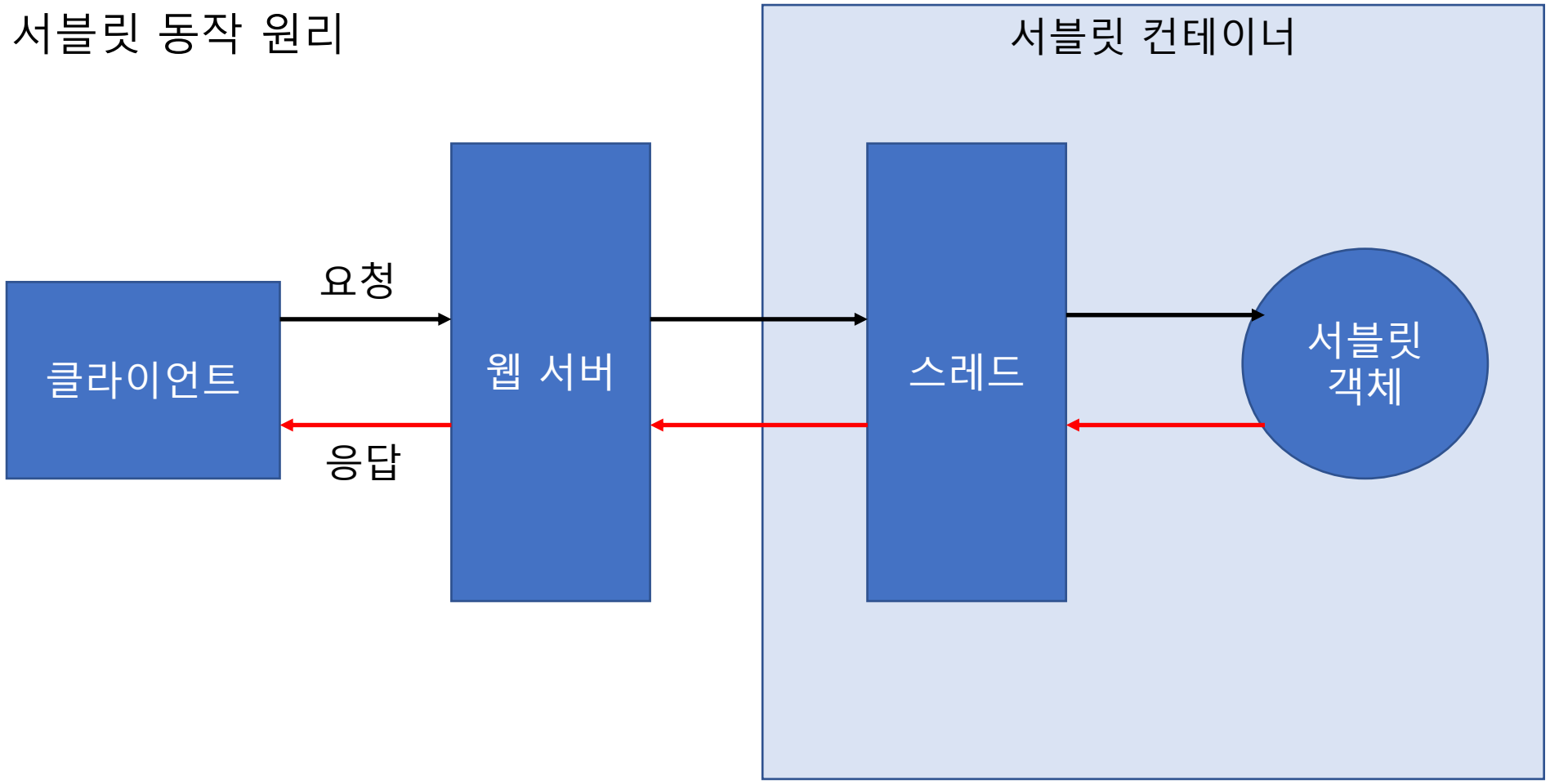
setContentType메소드는 응답할 페이지의 환경을 설정해 준다.
가장이 기본이 되는 응답방식 text/html
문자 인코딩 방식 charset=UTF-8

getWriter() 결과 출력을 위한 출력 스트림을 만들어 준다.

출력 스트림으로 만들어진 문자열을 응답해준다.

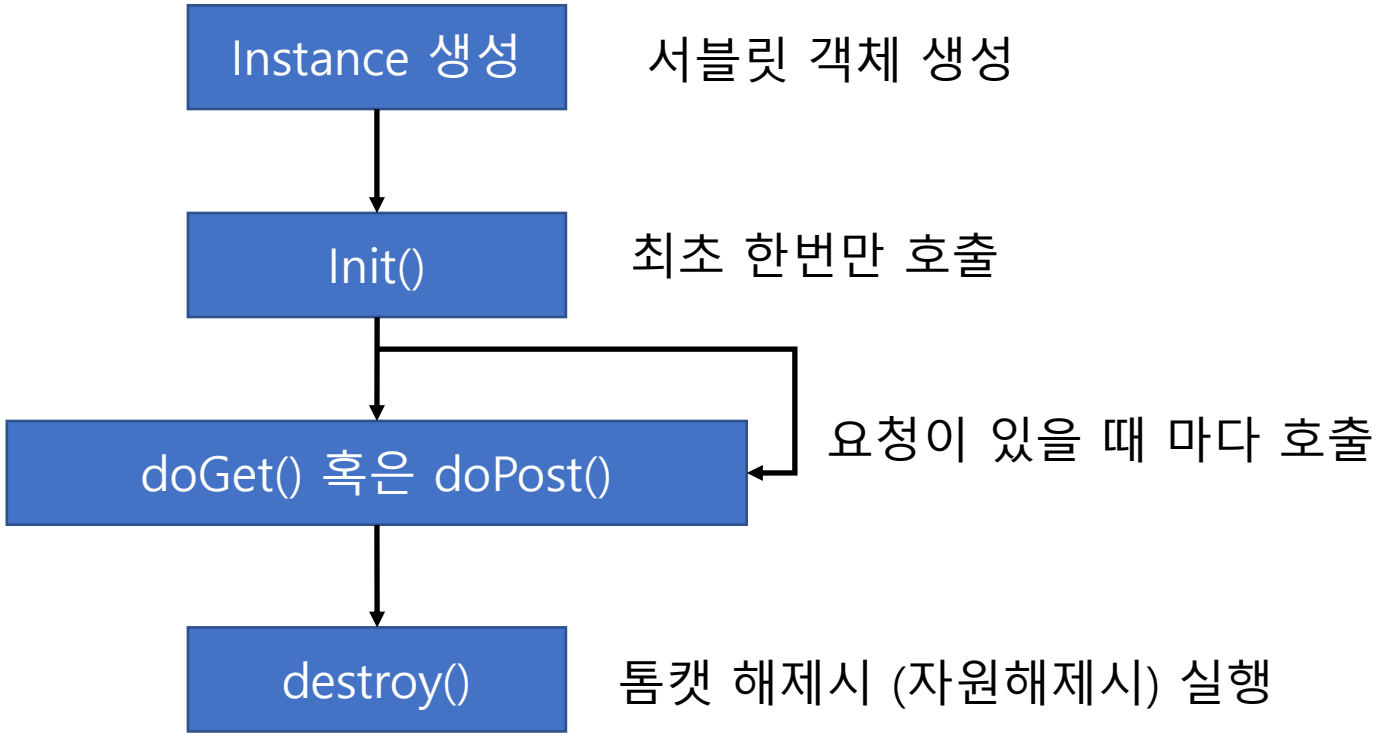
서블릿 프로그램을 만들어 보자

- 서블릿 동작 원리



서블릿 프로그램을 만들어 보자

- 서블릿의 라이프 사이클

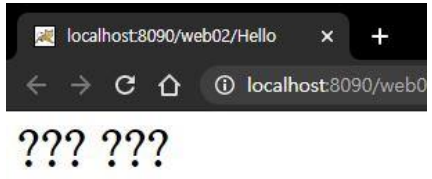


서블릿의 한글 처리

- 좀 전의 예제에서 'Hello Servlet'을 '헬로우 서블릿'으로 바꿔보자

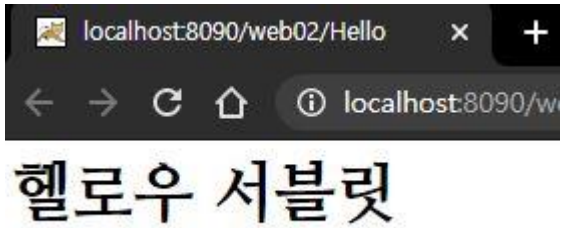
```
response.setContentType("text/html");
PrintWriter out = response.getWriter();
out.print("<html><body><h1>");
out.print("헬로우 서블릿");
out.print("</h1></body></html>");
out.close();
```

- 다음 처음 글자가 깨지는 현상을 볼 수 있다.



- 해결하기 위해서는 서블릿에 출력 인코딩 타입을 UTF-8을 적용하면 된다.

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



데이터 통신

- 클라이언트로부터 정보를 전달 받기 위해 폼태그를 사용한다.

- <form method="get/post" action="호출할 서블릿">

- form 태그의 주요 속성은 두가지 이다.
  - method : 정보를 어떤 방식으로 전달 할지 결정 Get, Post 두가지중 선택
  - action : 해당 정보를 어디에 전달 할지 결정..서블릿에서는 서블릿 매핑 주소를 명기한다.

- form태그에 적용된 내용을 전달하기 위해서는 반드시 전송 버튼이 있어야 한다.
  - <input type="submit" value="전송">

데이터 통신

- Get 전송 예제

```
<form method="get" action="MethodServlet">
        <input type="submit" value="get 방식으로 호출하기">
</form>
```

- POST 전송 예제

```
<form method="post" action="MethodServlet">
       <input type="submit" value="post 방식으로 호출하기">
</form>
```

- 서블릿

```
protected void doGet(HttpServletRequest request, HttpServ
    response.setContentType("text/html; charset=UTF-8");
    PrintWriter out = response.getWriter();
    out.print("<h1>get 방식으로 처리됨</h1>");
    out.close();
}
```

```
protected void doPost(HttpServletRequest request, HttpSe
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    out.print("<h1>post 방식으로 처리됨</h1>");
    out.close();
}
```

데이터 통신 – 쿼리 스트링

- 앞서 본 예는 클라이언트가 정보를 보내는 방식에 대해 알아보았다면 반대로 서버에서 정보를 읽어오는 방법에 대해 알아보자

- 서버에서는 클라이언트가 보낸 정보를 읽어 들이기 위해서 쿼리 스트링이라는 기술을 사용한다.

- 쿼리 스트링은 '리소스?이름=값& 이름=값'형태로 구성된다.

- 쿼리스트링을 사용하는 이유는 웹 페이지의 특성 때문인데 웹 페이지는 특성상 현재 페이지의 데이터를 다음 페이지가 인지 하지 못하고 전부 잃어버리기 때운이다.
- 그래서 페이지간의 정보를 교환할 필요를 쿼리스트링을 통해서 처리한다.

데이터 통신 – 쿼리 스트링

- 클라이언트에서 서버로 데이터를 전송하기 위해서는 기본적으로 데이터를 입력할 수 있는 텍스트 박스가 필요하다.

```
<input type="text" name="id">
<input type="text" name="age">
```

- 위 데이터는 서버로 데이터를 전송하는 데 name부분의 데이터를 '이름' 입력된 값을 '값' 형태의 쿼리 스트링 형태로 전송된다.

- 서버에서는 쿼리 스트링으로 전송되는 데이터를 어떻게 읽어 드릴까?

데이터 통신 – 쿼리 스트링

- 서버에서는 클라이언트로부터 오는 모든 데이터를 요청 객체(request)에 담아서 활용한다.

- 필요한 데이터는 request객체로 부터 얻을 수 있다.

```
request.getParameter('이름');
```

- 위 메소드로 얻어온 데이터는 문자열 형태로 사용가능 하다.

- 숫자가 필요하다면 문자열을 숫자로 변환해서 사용한다.

- 예제 작성

데이터 통신 – 쿼리 스트링

- 입력시 한글 처리

get방식으로 데이터 입력시 한글이 깨지는 경우
-server.xml 파일을 수정한다.

```
<Connector connectionTimeout="20000" port="8090"
    URIEncoding="UTF-8"
protocol="HTTP/1.1" redirectPort="8443"/>
```

Post 방식으로 데이터 입력시 한글이 깨지는 경우
doPost 메소드에 다음과 같은 코드를 추가한다,.

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

기타 다양한 입력 양식

- form 태그는 일반 텍스트 뿐 아니라 다양한 형식의 데이터를 입력받는다.
- 암호
- 라디오 버튼
- 체크박스
- 목록 상자

- 여러 데이터를 동시에 받는 경우 배열 형태로 받는다.