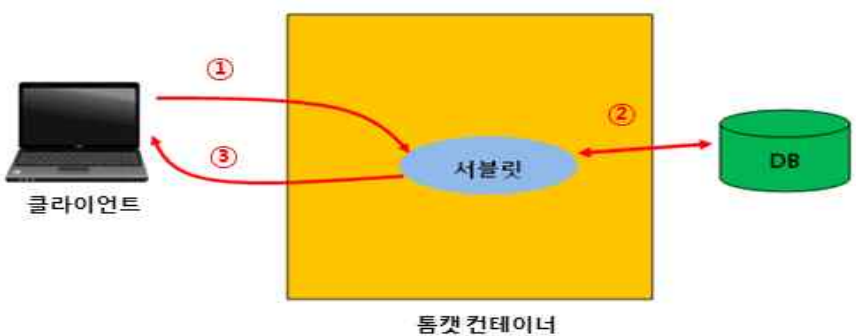
제3장
Servlet-Ⅱ

1. Servlet 작동 방식

➢ 초기의 웹 프로그래밍에선 서블릿을 이용해서 브라워저의 요청을 처리해서 서비스를 제공했음

- 서블릿의 세 가지 기본 기능



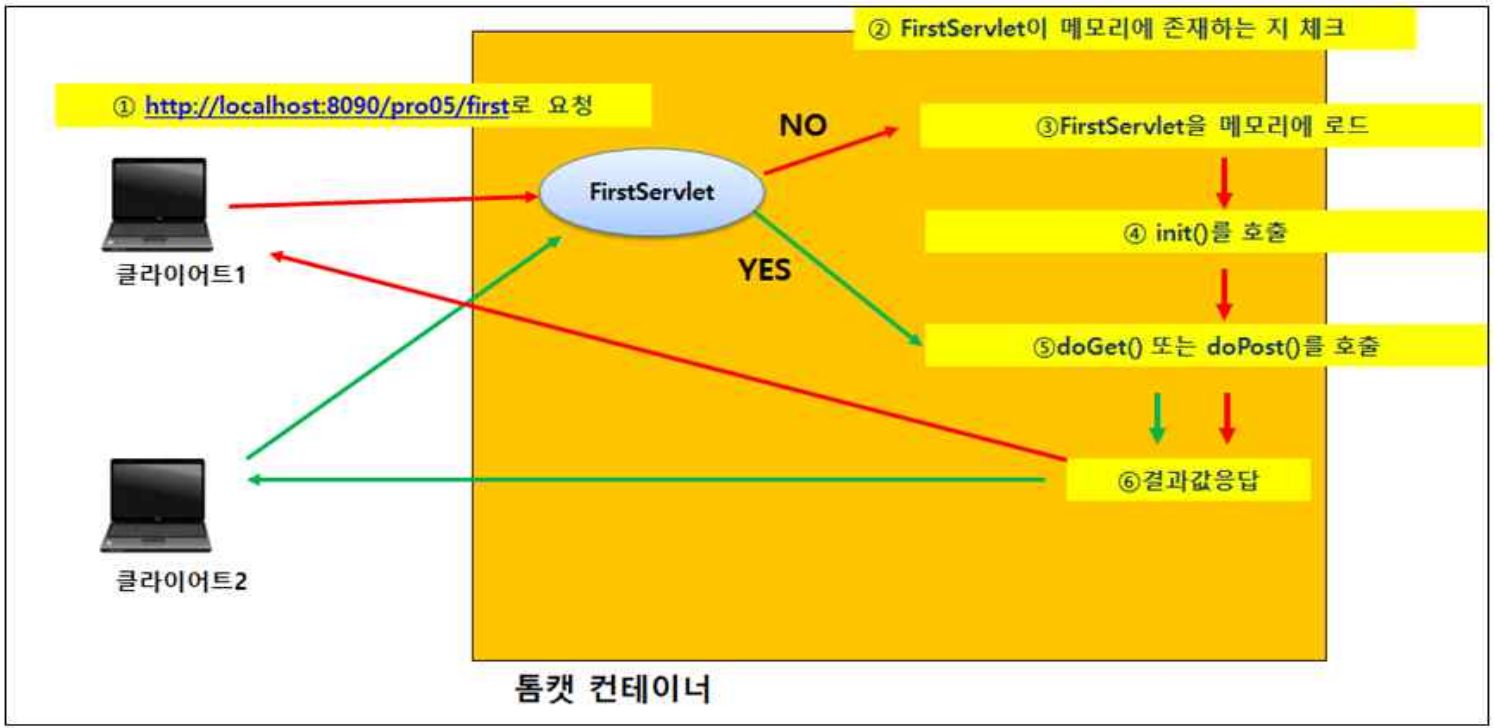
① 클라이언트로부터 요청을 얻음
② 데이터베이스 연동과 같은 비즈니스 로직을 처리함
③ 처리된 결과를 클라이언트에 응답

1. Servlet 작동 방식

클라이언트에서 servlet요청이 들어오면 서버에서는 servlet컨테이너를 만들고, 요청이 있을 때마다 스레드가 생성 됩니다.

자바가 다른 CGI(common gateway interface)언어에 비해 장점은 클라이언트에게서 계속적으로 요청이 들어오면, JVM이 스레드와 Servlet객체를 생성하여 서버의 부하가 심하지 않고 빠른 응답속도를 지니고 있다. 자바는 멀티스레드를 지원하기 때문이다. 이에 반해, 다른 CGI언어(perl)는 스레드 생성이 아니라 요청 처리를 위한 객체가 메모리에 지속적으로 상주하기 때문에 서버의 부하가 많이 심해진다. 하여, 보조프로그램을 같이 연동해서 사용하곤 한다.

1. Servlet 작동 방식



1. Servlet 작동 방식

- 요청과 관련된 API: javax.servlet.http.HttpServletRequest 클래스
- 응답과 관련된 API: javax.servlet.http.HttpServletResponse 클래스

- 요청과 응답 관련 API 사용 예

```
public class FirstServlet extends HttpServlet {
    @Override
    public void init() throws ServletException {
        System.out.println("init 메서드 호출");
    }

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

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        System.out.println("doGet 메서드 호출");
    }
    @Override
    public void destroy() {
        System.out.println("destroy 메서드 호출");
    }
}
```

1. Servlet 작동 방식

HttpServletRequest의 여러가지 메서드

반환형	메서드 이름	기능
boolean	Authenticate (HttpServletResponse response)	현재 요청한 사용자가 ServletContext 객체에 대한 인증을 하기 위한 컨테이너 로그인 메커니즘을 사용합니다.
String	changeSessionId()	현재 요청과 연관된 현재 세션의 id를 변경하여 새 세션 id를 반환합니다.
String	getContextPath()	요청한 컨텍스트를 가리키는 URI를 반환합니다.
Cookie[]	getCookies()	클라이어트가 현재의 요청과 함께 보낸 쿠키 객체들에 대한 배열을 반환합니다.
String	getHeader(String name)	특정 요청에 대한 헤더 정보를 문자열로 반환합니다.
Enumeration <String>	getHeaderNames	현재의 요청에 포함된 헤더의 name 속성을 enumeration으로 반환합니다.
String	getMethod()	현재 요청이 GET, POST 또는 PUT 방식 중 어떤 HTTP 요청인지 를 반환합니다.
String	getRequestURI()	요청한 URL의 컨텍스트 이름과 파일 경로까지 반환합니다.
String	getServletPath()	요청한 URL에서 서블릿이나 JSP 이름을 반환합니다.
HttpSession	getSession()	현재의 요청과 연관된 세션을 반환합니다. 만약 세션이 없으면 새로 만들어서 반환합니다.
String	getPathInfo()	클라이언트가 요청 시 보낸 URL과 관련된 추가 경로 정보를 반환합니다.

1. Servlet 작동 방식

HttpServletResponse의 여러가지 메서드

반환형	메서드 이름	기능
void	addCookie (Cookie cookie)	응답에 쿠키를 추가합니다.
void	addHeader(String name, String value)	name과 value를 헤더에 추가합니다.
String	encodeURL(String url)	클라이언트가 쿠키를 지원하지 않을 때 세션 id를 포함한 특정 URL을 인코딩합니다.
Collection <String>	getHeaderNames()	현재 응답의 헤더에 포함된 name을 얻어옵니다.
void	sendRedirect (String location)	클라이언트에게 리다이렉트(redirect) 응답을 보낸 후 특정 URL로 다시 요청하게 합니다.

2. Servlet 라이프사이클(생명주기)

Servlet의 사용도가 높은 이유는 빠른 응답 속도 때문입니다.
Servlet은 최초 요청 시 객체가 만들어져 메모리에 로딩되고, 이후 요청 시에는 기존의 객체를 재활용하게 됩니다.
따라서 동작 속도가 빠릅니다. Servlet의 라이프 사이클을 살펴 봅니다.

최초 한번

최초 한번

요청시 매번(중요)

마지막 한번
(자원 해제 : servlet 수정시 재컴파일, 서버 재가동 등등..)

Servlet객체는 클라이언트의 요청이 들어오면 생성이 되고 아울러, 그 생성된 객체를 재사용을 하기 때문에 응답속도 및 동작 속도가 빠르다.
아울러, Servlet객체가 생성되기 위해서는 java파일의 class파일이 존재해야 되는 것은 이미 잘 알고 있다.

이미 앞선 장에서 doGet()방식과 doPost()방식을 살펴보았다. 다시 한번 상기시키도록 하자.
doGet()은 폼태그에서 전송방식을 get으로 하면 doGet()이 실행되고 또한 브라우저에 바로 입력하면 doGet()이 실행된다. 반면에 post는 폼태그에서 post로 전달되어야 doPost()가 실행된다.

service()는 doGet()과 doPost()가 없을 때, 자동 실행되는 콜백 메서드이다. 하지만 doGet(),doPost()가 있다 하더라도 service()만 실행한다. 하지만 통상 service()는 잘 사용하지 않는다.

3. Servlet 선처리, 후처리

Servlet의 라이프 사이클 중 init()과 destroy()메소드와 관련하여 선처리(init()전)와 후처리(destroy()후) 작업이 가능합니다.

4. HTML form태그

Html의 form태그는 서버 쪽으로 정보를 전달할 때 사용하는 태그이다.
Html의 모든 태그를 학습할 필요는 없습니다. 하지만 웹 프로그래머로서 Html언어를 어느 정도는 할 수 있어야 한다.
틈틈이 공부하자.

input

태그의 종류를 지정한다.

속성(type, name, value)
- type : 태그 종류 지정(ex. text, password(비밀번호를 보이지 않게), submit(데이터 전송시), checkbox(네모버튼, 다중선택가능), radio(동그란 선택, 1개만 선택함), reset(정보입력 후 다시 다 지우고 싶을때 사용))
- name : input태그 이름(servlet으로 전송할 때의 이름, 역시 servlet에서도 같은 이름을 사용할 수 있다)
- value : name에 해당하는 값(ex. name = value(보이는 값))

type = text

일반적인 데이터를 입력하기 위해 사용한다.

```
<input type="text" name="name" size="10">
```

type = password

로그인, 회원가입 페이지 등에서 비밀번호 입력하기 위해 사용한다.

```
<input type="password" name="name" size="10">
```

4. HTML form태그

type = submit(버튼으로 표시가 됨)

form내의 데이터를 서버쪽으로 전송할 때 사용한다.

```
<input type="submit" value="전송">
```

type = reset

form내의 데이터를 초기화 할 때 사용한다.

```
<input type="reset" value="초기화">
```

type = checkbox

데이터 값을 여러 개 전송해야 할 때 사용한다.(다중 선택)

```
<input type="checkbox" name="hobby" value="read">독서
<input type="checkbox" name="hobby" value="cook">요리
<input type="checkbox" name="hobby" value="run">조깅
<input type="checkbox" name="hobby" value="swim">수영
<input type="checkbox" name="hobby" value="sleep">취침
```

실제 서버에 전송되는 값은 value의 값이다.

4. HTML form태그

type = radio

checkbox와 달리 여러 개의 데이터 값 중 한 개의 값만을 전송할 때 사용합니다.

```
<input type="radio" name="major" value="kor">국어
<input type="radio" name="major" value="eng" checked="checked">영어->checked는 디폴트 값
<input type="radio" name="major" value="mat" >수학
<input type="radio" name="major" value="des" >디자인
```

select

리스트형태의 데이터를 사용합니다.(하나만 선택)

```
<select name="protocol">
    <option value="http">http</option>
    <option value="ftp" selected="selected">ftp</option>
     <option value="smtp">smtp</option>
     <option value="pop">pop</option>
</select>
```

4. HTML form태그

form 태그

input 태그들의 값을 서버로 전송하기 위한 정보를 담고 있습니다.

```
<form action="FormEx" method="post">
```

요청하는 컴포넌트 이름
(form태그실행 시 찾아갈 소스)
(ex. join.jsp, info.html, HWorld)

요청을 처리하는 방식
(ex. get, post)

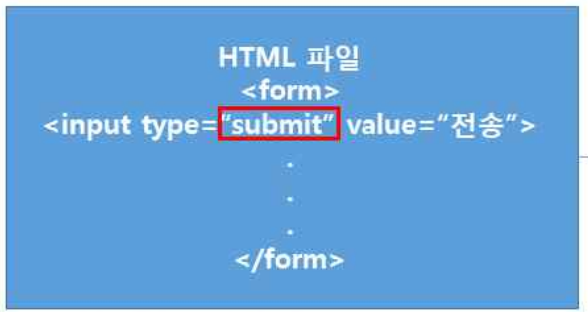
Get : http://IP주소:port번호/컨텍스트/path/MemberJoin?id=abcdefg&name=홍길동

Post : http://IP주소:port번호/컨텍스트/path/MemberJoin

** 앞선, 강의에서도 설명했지만 Get방식은 위와 같이 아이디나 비번 등 값을 주소표시줄에 표식을 하기 때문에 보안에 취약하다. 반면에 Post방식은 헤더에 값들을 실어서 전송하기 때문에 보안에 강하다.

4. HTML form태그

form태그의 submit 버튼을 클릭하여 데이터를 서버로 전송하면, 해당파일(Servlet)에서는 HttpServletRequest객체를 이용하여 Parameter값을 얻을 수 있다.




```
<form action="FormEx" method="post">
    이름 : <input type="text" name="name" size="10"><br/>
    아이디 : <input type="text" name="id" size="10"><br/>
    비밀번호 : <input type="password" name="pw" size="10"><br/>
```

```
protected void doPost(HttpServletRequest request, HttpServ
    // TODO Auto-generated method stub
    System.out.println("doPost");

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

    response.setContentType("text/html; charset=EUC-KR");
    PrintWriter writer = response.getWriter();

    writer.println("<html><head></head><body>");
    writer.println("아이디 : " + id + "<br />");
    writer.println("비밀번호 : " + pw + "<br />" );
```

아이디 : abcdefg
비밀번호 : 1234

1. getParameter(name) : 해당 name의 value값을 리턴해 줌
2. getParameterValues(name) : 해당 name의 value값을 리턴해 주는데, 값이 여러개인 것 즉, 체크박스의 값을 리턴해준다.
3. getParameterNames() : form태그에서 넘어온 name값들을 전부 배열로 리턴 해준다.

5. 한글 처리

Tomcat 서버의 기본 문자 처리 방식은 IOS-8859-1 방식이다. 따라서 개발자가 별도의 한글 인코딩을 하지 않으면 한글이 깨져 보이는 현상이 있다.
Get방식과 Post방식에 따라서 한글처리 방식에 차이가 있다.

Get방식 요청

<server.xml 수정>

```
<Connector URIEncoding= "UTF-8" port="8181" cor
```

Get방식의 한글깨짐 방지는 Server의 server.xml파일은 톰캣서버의 복사본 형태이다. 하여, 위와 같이 추가 한 후, Publish to the Server를 버튼을 눌러야 서버와 동기화가 된다. 단축키로 Ctrl+Alt+P를 사용해도 좋다.

Post방식 요청

<request.setCharacterEncoding() 메소드 이용>

```
protected void doPost(HttpServletRequest reques
    // TODO Auto-generated method stub
    System.out.println("doPost");

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

post방식은 직접 자바코드로 위와 같이 입력해주면 된다.

감사합니다.