

JSP

– Servlet

1. URL맵핑
2. 요청방식(Get, Post)
3. 한글처리

Servlet

JAVA 코드

* Servlet 특징

1. 동적 웹어플리케이션 컴포넌트 (순수 자바코드)
2. .java 확장자
3. 클라이언트의 요청에 동적으로 작동하고, 응답은 html을 이용.
4. java thread를 이용하여 동작.
5. MVC패턴에서 Controller로 이용됨.

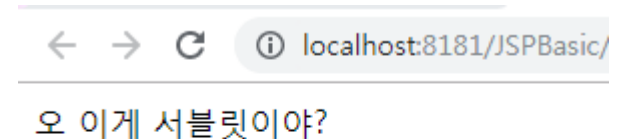
* Servlet 특징

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {

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

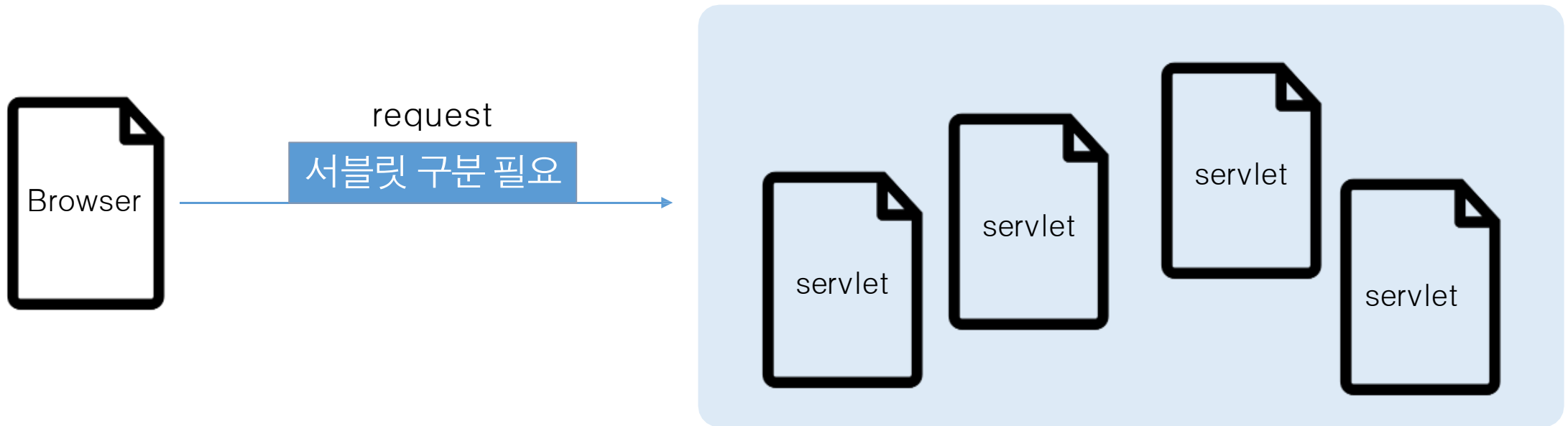
    PrintWriter out = response.getWriter();
    out.println("<body>");
    out.println("오 이게 서블릿이야?");
    out.println("</body>");

}
```



서블릿은 여러 개가 있을 수 있다!!

웹 컨테이너(tomcat)



서블릿 구분 방법

1. URL-Mapping

- URL 매핑을 하지 않으면 URL주소가 너무 길어지고, 경로가 노출되어 보안에 위험이 생기기 때문에 URL 매핑을 사용하여 그 문제들을 해결합니다.

- http://localhost:8181/JSPBasic/servlet/kr.co.park.HelloWorld
---->> http://localhost:8181/JSPBasic/HelloWorld

- 사용 방법

1. 어노테이션 이용, 클래스 선언부 바로 위에 작성.

ex) @WebServlet("/HelloWorld")

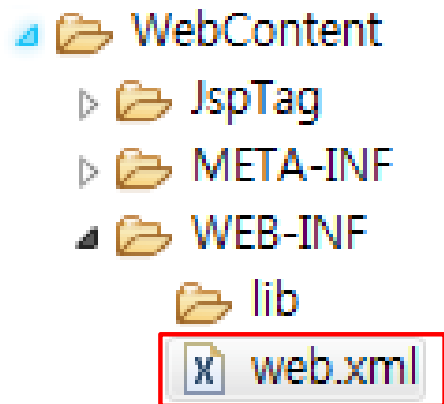
```
@WebServlet("/banana")
```

```
public class ServletBasic extends HttpServlet {
```

```
}
```

서블릿 구분 방법

2. web.xml 설정파일 수정.



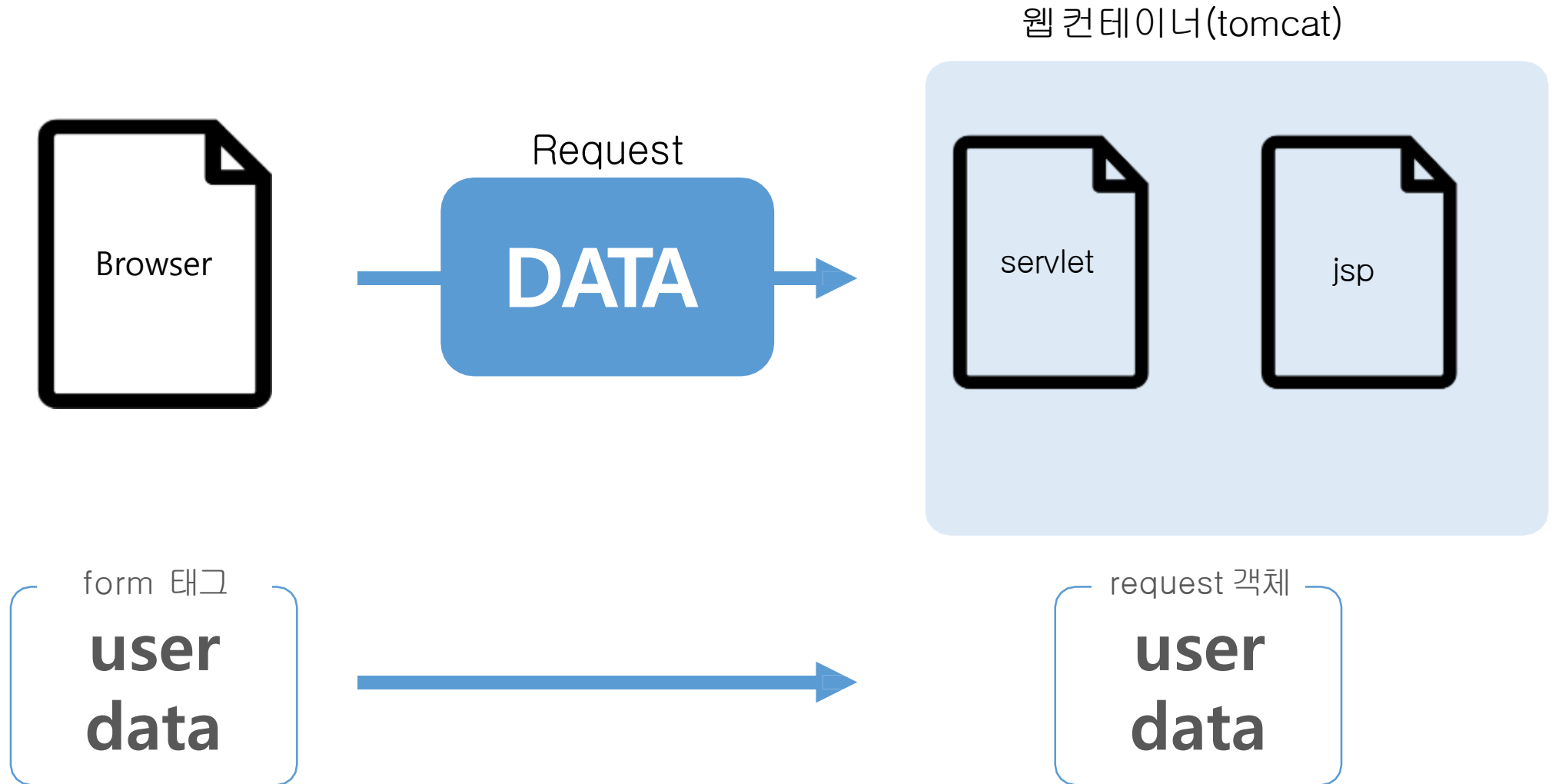
웹 프로젝트 설정 파일



```
<servlet>
  <servlet-name>basic</servlet-name>
  <servlet-class>kr.co.koo.ServletBasic</servlet-class>
</servlet>

<servlet-mapping>
  <servlet-name>basic</servlet-name>
  <url-pattern>/peach</url-pattern>
</servlet-mapping>
```

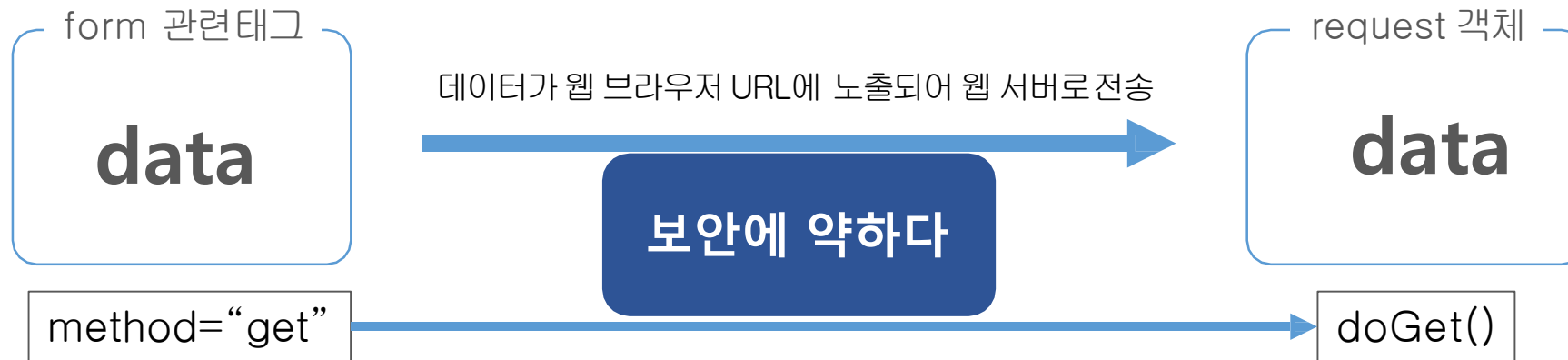
* HttpRequest 방식 2가지



* HttpRequest 방식 2가지

- GET 방식

1. 서버에 데이터를 요청하는 용도.
2. 전송하는 **데이터가 주소**에 묻어서 감.
3. 전송했던 데이터는 브라우저의 히스토리에 접속했던 주소와 함께 남아 있어 **보안성에 취약함**.
4. 게시판 글 조회나 검색 같이 서버의 정보를 가져올 필요성이 있을 때 사용함.
5. 전송할 수 있는 최대 크기는 브라우저별로 다르지만 크기가 정해져 있음.
6. HTML **form태그** 가 반드시 필요하지는 **않습니다**.

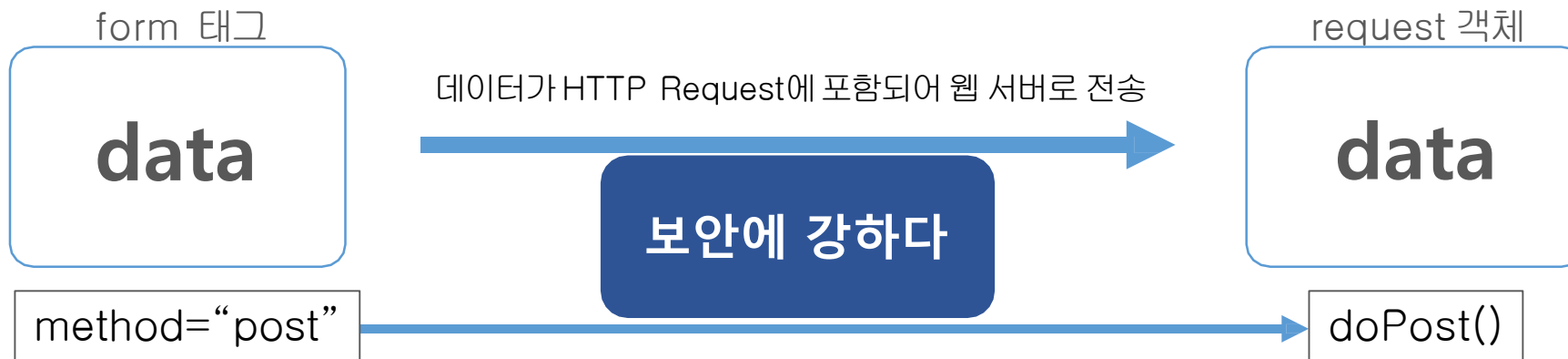


URL 주소를 확인해보자!

* HttpRequest 방식 2가지

- POST 방식

1. 서버에 데이터를 전송하는 용도.
2. 전송되는 데이터가 URL에 묻어나가지 않고 전송 객체의 메시지 바디를 통해 전달됨.
3. 브라우저에 전달되는 데이터가 남지 않기 때문에 **보안성에 강함**.
4. 비밀번호나 주민번호 등 private한 데이터를 서버에 전송해야 할 때 사용함.
5. 반드시 HTML **form태그**가 필요합니다.
6. 데이터 양의 제한이 없기 때문에 대량의 데이터를 전송할 수 있습니다.



URL 주소를 확인해보자!

* GET/ POST 방식 브라우저 한글처리

- 톰캣서버의 기본 문자처리 방식은 IOS-8859-1 방식입니다.
- 따라서 개발자가 별도의 한글 인코딩을 하지 않으면 서버로 전송된 데이터의 한글들이 깨져보이는 현상이 발생합니다.

1. GET 방식의 한글처리

- server.xml 파일 수정
- <connector> 에 속성 값으로 URLEncoder="utf-8"



```
server.xml
59      Java AJP  Connector: /docs/config/ajp.1
60      APR (HTTP/AJP) Connector: /docs/apr.ht
61      Define a non-SSL/TLS HTTP/1.1 Connecto
62      -->
63      <Connector URLEncoder="UTF-8" connectionTir
64      <!-- A "Connector" using the shared thread
65      <!--
66      <Connector name="http" thread="1" />
```

2. POST 방식의 한글처리

- post 방식을 처리하는 메서드에

request.setCharacterEncoding("utf-8");



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

    protected void doGet(HttpServletRequest request, HttpServlet
        request.setCharacterEncoding("UTF-8");
        response.setContentType("text/html; charset=UTF-8");
    }
}
```

```
mSignUp.jsp
1 <% request.setCharacterEncoding("UTF-8"); %>
2 <%@ page language="java" contentType="text/html; c
3     pageEncoding="UTF-8"%>
4 <!DOCTYPE html>
5 <html>
6 <head>
7 <meta charset="UTF-8">
```