* HttpRequest 방식
- GET 방식
1. 서버에 데이터를 요청하는 용도.
2. 전송하는 데이터가 주소에 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 정해져있음.
6. HTML form태그가 반드시 필요하지는 않습니다.

- 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 form태그가 필요합니다.
6. 데이터 양의 제한이 없기 때문에 대량의 데이터를 전송할 수 있습니다.

* 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 URIEncoding="utf-8"

2. POST 방식의 한글처리
- post 방식을 처리하는 메서드(스크립트릿)에
request.setCharacterEncoding("utf-8");

* response 객체의 이해
- 웹 브라우저의 요청에 응답하는 것을 response라고 합니다.
- 이러한 응답의 정보를 가지고 있는 객체를 response객체라고 합니다.

- response 객체 주요 메서드
1. getCharacterEncoding(): 응답할 때의 문자의 인코딩 형태를 구합니다.
2. addCookie(Cookie c): 쿠키를 지정합니다.
3. sendRedirect(URL): 지정한 URL로 이동합니다.

* out 객체의 이해
- JSP 페이지가 생성하는 모든 내용은 out 기본 객체를 통해 전송됩니다.
- JSP 페이지 내에서 사용하는 비스크립트 요소들(HTML코드와 텍스트)이 out 객체에 전달됩니다.
- 값을 출력하는 표현식(expression)의 결과값도 out객체에 전달됩니다.

* Servlet 특징
1. 동적 웹어플리케이션 컴포넌트
2. .java 확장자
3. 클라이언트의 요청에 동적으로 작동하고, 응답은 html을 이용.
4. java thread를 이용하여 동작.
5. MVC패턴에서 Controller로 이용됨.

* 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.koo.HelloWorld
---->> http://localhost:8181/JSPBasic/HelloWorld

- 사용 방법
1. 아노테이션 이용, 클래스 선언부 바로 위에 작성.
ex) @WebServlet("/HelloWorld")

2. web.xml 설정파일 수정.
ex)
```
<servlet>
        <servlet-name>helloworld</servlet-name>
        <servlet-class>kr.co.koo.HelloWorld</servlet-class>
  </servlet>
  <servlet-mapping>
        <servlet-name>helloworld</servlet-name>
        <url-pattern>/HWorld</url-pattern>
  </servlet-mapping>
```

* Servlet 작동 순서

- 클라이언트에서 요청(request)이 들어오면 서버에서는 servlet 컨테이너를 만들고, 요청이 있을때 thread와 Servlet 객체가 생성됩니다.

* Servlet의 생명주기(LifeCycle)
- Servlet의 장점은 빠른 응답 속도입니다.
- Servlet은 최초 요청시에 객체가 만들어져 메모리에 로딩되고, 이후 추가 요청시에는 기존의 객체를 재활용하게 됩니다. 따라서 동작 속도가 매우 빠릅니다.

1. Servlet 객체를 생성 (최초 한번)
2. Init() 메서드 호출(최초 한번)
3. doGet(), doPost(), service() 호출 (요청시 매번)
4. destroy() 호출 (마지막 한번) - 자원이 해제될 시 호출(Servlet코드를 수정, 서버 재가동할 시)

* 웹 어플리케이션 생명주기 (ServletContextListener)
- 웹 어플리케이션에는 프로그램의 생명주기를 감시하는 리스너가 있습니다.
- 리스너의 해당 메서드가 웹 어플리케이션의 시작과 종료시에 호출됩니다.
1. 시작시에는 contextInitialized()
2. 종료시에는 contextDestroyed()

* 서블릿 초기화 파라미터 (ServletConfig)
- 특정한 서블릿이 생성될 때 초기에 필요한 데이터들이 있습니다.
- 이러한 데이터들을 초기화 파라미터라고 하며, 아노테이션으로 지정하는 방법과, web.xml파일에 기술하는 방법이 있습니다.

* 데이터 공유(ServletContext)
- 여러 서블릿에서 특정 데이터를 공유해야 할 경우 Context Parameter를 이용하여 web.xml파일에 데이터를 기술하고, 여러 서블릿에서 공유하면서 사용할 수 있습니다.