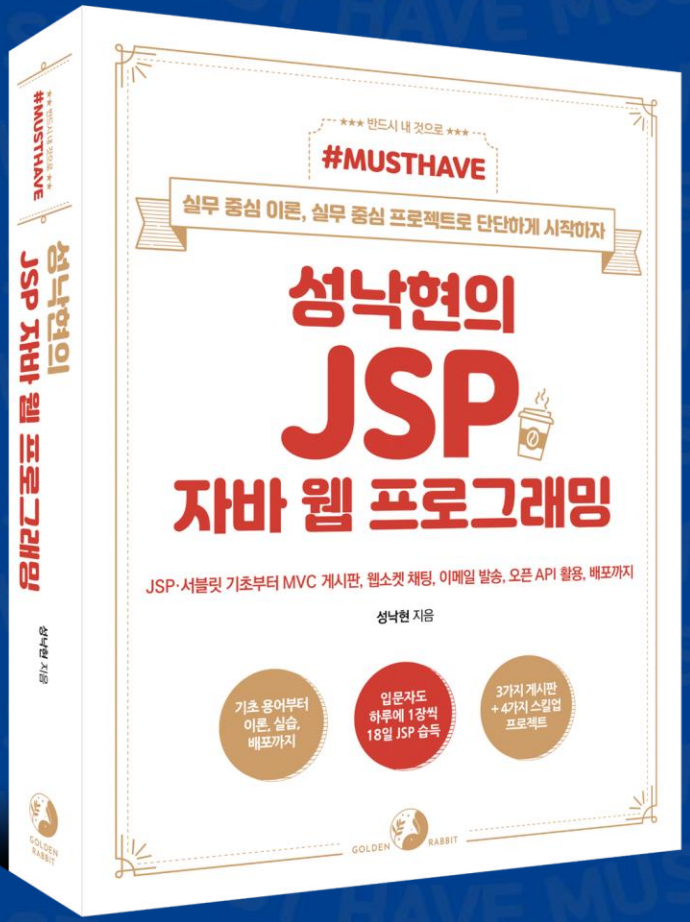


Chapter

02

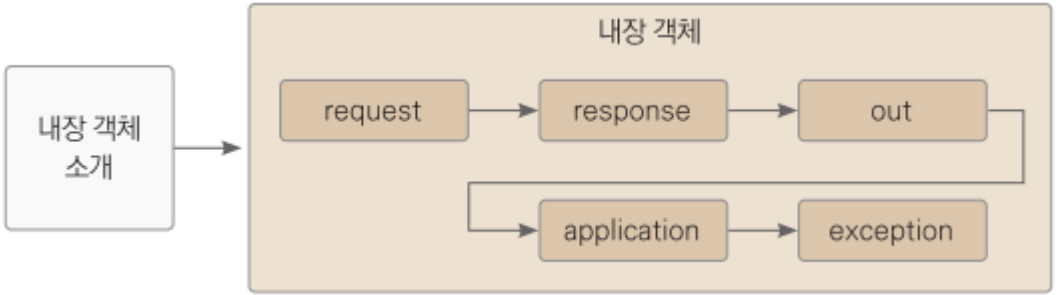
내장 객체 (Implicit Object)

학습 목표 및 순서

▣ 학습 목표

- 클라이언트의 요청을 받거나 응답할 때 사용되는 JSP의 기본 내장 객체들의 종류와 사용법을 익힘.

▣ 학습 순서



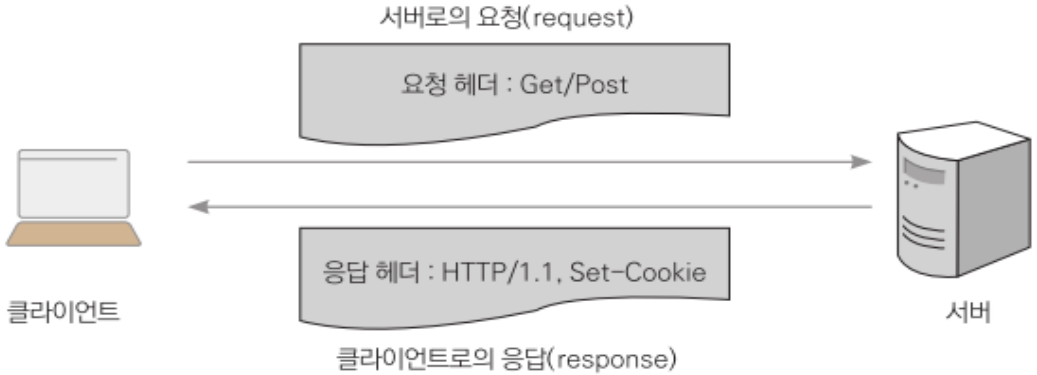
▣ 활용 사례

- 요청과 응답부터 출력, 세션, 페이지와 애플리케이션 등 없어서는 안 될 개념들을 내장 객체로 제공하므로 수시로 광범위하게 활용됩니다.

2.1 내장 객체란?

▣ 내장객체

- JSP의 내장 객체(Implicit Object)는 기본적인 요청과 응답, 화면출력 등을 담당



- Tomcat 컨테이너가 미리 선언하여 별도의 생성없이 사용할 수 있음
- <% 스크립틀릿 %>과 <%= 표현식 %>에서는 즉시 사용 가능
- <%! 선언부 %>에서는 매개변수로 전달받아 사용할 수 있음

이유는..??

2.1 내장 객체란?

- 내장객체는 JSP가 서블릿으로 변환될때 생성되는 _jspService( ) 메서드 내부에 선언되기 때문

```
public void _jspService() {
    ... 생략 ...
    final jakarta.servlet.jsp.PageContext pageContext;
    jakarta.servlet.http.HttpSession session = null;
    final jakarta.servlet.ServletContext application;
    final jakarta.servlet.ServletConfig config;
    jakarta.servlet.jsp.JspWriter out = null;
    final java.lang.Object page = this;
    jakarta.servlet.jsp.JspWriter _jspx_out = null;
    jakarta.servlet.jsp.PageContext _jspx_page_context = null;
    ... 생략 ...
```

- 스크립틀릿과 표현식의 코드는 _jspService( ) 내부에 기술됨 ⇒ 즉시 사용 가능
- 선언부의 코드는 _jspService( ) 외부에 기술됨 ⇒ 매개변수로 전달 받은 후 사용 가능

2.1 내장 객체란?

▣ 내장객체의 종류

내장 객체	타입	설명
request	jakarta.servlet.http.**HttpServletRequest**	클라이언트의 요청 정보를 저장합니다.
response	jakarta.servlet.http.**HttpServletResponse**	클라이언트의 요청에 대한 응답 정보를 저장합니다.
out	jakarta.servlet.jsp.**JspWriter**	JSP 페이지에 출력할 내용을 담는 출력 스트림입니다.
session	jakarta.servlet.http.**HttpSession**	웹 브라우저 정보를 유지하기 위한 세션 정보를 저장합니다.
application	jakarta.servlet.**ServletContext**	웹 애플리케이션 관련 컨텍스트 정보를 저장합니다.
pageContext	jakarta.servlet.jsp.**PageContext**	JSP 페이지에 대한 정보를 저장합니다.
page	java.lang.**Object**	JSP 페이지를 구현한 자바 클래스의 인스턴스입니다.
config	jakarta.servlet.**ServletConfig**	JSP 페이지에 대한 설정 정보를 저장합니다.
exception	java.lang.**Throwable**	예외가 발생한 경우에 사용합니다.

2.2 request 객체

▣ request 객체

- JSP에서 가장 많이 사용되는 객체로, 클라이언트의 요청 정보를 담고 있음
- 주요기능
  - 클라이언트와 서버에 대한 정보 읽기
  - 클라이언트가 전송한 요청 매개변수에 대한 정보 읽기
  - 요청 헤더 및 쿠키 정보 읽기

▣ 클라이언트와 서버의 환경정보 읽기

- 클라이언트의 요청에 따른 전송방식(GET / POST)
- 요청 URL(HOST포함) 및 URI(HOST 미포함)
- 포트번호, IP주소 등의 정보를 얻어올 수 있음

예제 2-1] 02ImplicitObject/RequestMain.jsp
예제 2-2] 02ImplicitObject/RequestWebInfo.jsp

2.2 request 객체

▣ 클라이언트의 요청 매개변수 읽기

- <form> 태그 하위 요소를 통해 전송(submit)한 폼값을 받을 수 있음
- 전송된 값이 하나인 경우 getParameter() 사용
  - <input> 태그의 type속성이 text, password, radio 일때 입력값
  - <textarea> 태그의 입력값
  - <select> 태그의 선택값(multiple 속성 없음)
- 전송된 값이 둘 이상인 경우 getParameterValues() 사용
  - <input> 태그의 type속성이 checkbox 일때 선택값
  - <select> 태그의 여러 선택값(multiple 속성 추가)

예제 2-3] 02ImplicitObject/RequestParameter.jsp

```
<body>
<%
request.setCharacterEncoding("UTF-8");  ❶
String id = request.getParameter("id");  ❷
String sex = request.getParameter("sex");
String[] favo = request.getParameterValues("favo");  ❸
```

2.2 request 객체

▣ HTTP 요청 헤더 정보 읽기

- HTTP 프로토콜은 헤더에 부가적인 정보를 저장할 수 있음
  - user-agent : 웹 브라우저의 종류
  - referer : 웹을 서핑하면서 링크를 통해 다른 사이트로 방문 시 남는 흔적
  - cookie : 쿠키 정보
  - 그 외 여러가지 정보

예제 2-4] 02ImplicitObject/RequestHeader.jsp

```
<body>
    <h2>3. 요청 헤더 정보 출력하기</h2>
    <%
    Enumeration headers = request.getHeaderNames();  ❶
    while (headers.hasMoreElements()) {  ❷
        String headerName = (String)headers.nextElement();  ❸
        String headerValue = request.getHeader(headerName);  ❹
        out.print("헤더명 : " + headerName + ", 헤더값 : " + headerValue + "<br/>");
    }
    %>
    <p>이 파일을 직접 실행하면 referer 정보는 출력되지 않습니다.</p>
</body>
```

2.3 response 객체

▣ response 객체

- 클라이언트의 요청에 대한 응답을 웹 브라우저로 보내주는 역할
- 주요 기능
  - 페이지 이동을 위한 리다이렉트(redirect)
  - HTTP 헤더에 응답 헤더 추가
  - 그 외에는 거의 사용되지 않음

▣ sendRedirect()로 페이지 이동하기

- HTML의 <a> 태그, 자바스크립트의 location 객체와 동일
- 단순 문자열 비교를 통한 로그인 기능 구현

2.3 response 객체

예제 2-5] 02ImplicitObject/ResponseMain.jsp

예제 2-6] 02ImplicitObject/ResponseLogin.jsp

```
<body>
<%
String id = request.getParameter("user_id");
String pwd = request.getParameter("user_pwd");                    ❶
if (id.equalsIgnoreCase("must") && pwd.equalsIgnoreCase("1234")) {  ❷
    response.sendRedirect("ResponseWelcome.jsp");                   ❸
}
else {
    request.getRequestDispatcher("ResponseMain.jsp?loginErr=1")     ❹
       .forward(request, response);
}
%>
</body>
```

❶ 아이디/비번을 파라미터로 받음
❷ 단순 문자열 비교로 검증
❸ 인증성공인 경우 페이지 이동
❹ 인증실패인 경우 포워드

2.3 response 객체

예제 2-7] 02ImplicitObject/ResponseWelcome.jsp

인증에 실패한 경우에는 ResponseMain.jsp로 포워드

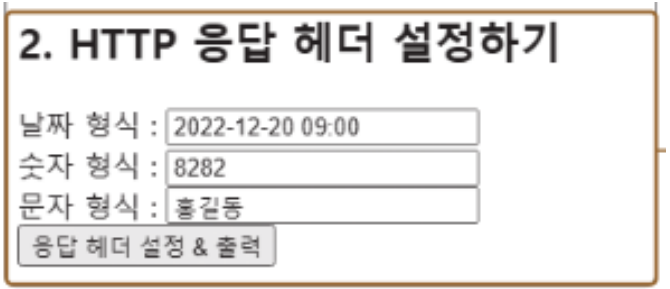
- 주소줄에는 ResponseLogin.jsp로 표시
- 하지만 화면에는 ResponseMain.jsp의 내용이 출력
- '포워드'는 이와같이 실행의 흐름만 특정 페이지로 넘겨주는 역할을 함

2.3 response 객체

▣ HTTP 헤더에 응답 헤더 추가하기

- response 내장 객체는 응답 헤더에 정보를 추가하는 기능을 제공
- add 계열은 헤더값을 새로 추가할 때 사용
- set 계열은 기존의 헤더를 수정할 때 사용

예제 2-5] 02ImplicitObject/ResponseMain.jsp



예제 2-8] 02ImplicitObject/ResponseHeader.jsp

```
<%
// 응답 헤더에 추가할 값 준비
SimpleDateFormat s = new SimpleDateFormat("yyyy-MM-dd HH:mm");
long add_date = s.parse(request.getParameter("add_date")).getTime();   ❶
int add_int = Integer.parseInt(request.getParameter("add_int"));   ❷
String add_str = request.getParameter("add_str");

// 응답 헤더에 값 추가
response.addDateHeader("myBirthday", add_date);
response.addIntHeader("myNumber", add_int);
response.addIntHeader("myNumber", 1004); // 추가 ❸
response.addHeader("myName", add_str);
response.setHeader("myName", "안중근");  // 수정 ❹
%>
```

❶ 날짜포맷은 시간까지 설정하고, 시간이 09:00 이후여야 오늘날짜로 설정됨.

❸ addXX() 로 헤더값 추가

❹ setXX() 로 기존 헤더값 수정

2.3 response 객체

▣ HTTP 헤더에 응답 헤더 추가하기

- getHeader() 메서드로 출력하면 값이 여러개라도 첫 번째 값만 가져옴

- getHeaders() 메서드로 출력하면 같은 헤더명이라도 다른 값을 출력함

2.4 out 객체

▣ out 객체

- 웹 브라우저에 변수 등의 값을 출력할 때 주로 사용
- 출력되는 모든 정보는 버퍼에 먼저 저장된 후 웹 브라우저에 출력

예제 2-9] 02ImplicitObject/OutMain.jsp

```
<%
// 버퍼 내용 삭제하기
out.print("출력되지 않는 텍스트");  // 버퍼에 저장 ❶
out.clearBuffer();  // 버퍼를 비움(윗 줄의 출력 결과 사라짐) ❷

out.print("<h2>out 내장 객체</h2>");

// 버퍼 크기 정보 확인
out.print("출력 버퍼의 크기 : " + out.getBufferSize() + "<br>");  ❸
out.print("남은 버퍼의 크기 : " + out.getRemaining() + "<br>");   ❹

out.flush();  // 버퍼 내용 출력 ❺
out.print("flush 후 버퍼의 크기 : " + out.getRemaining() + "<br>");  ❻
```

2.5 application 객체

▣ application 객체

- 웹 애플리케이션당 하나만 생성되며, 모든 JSP 페이지에서 접근할 수 있음
- 주요기능
  - web.xml에 설정한 컨텍스트 초기화 매개변수 읽기
  - 폴더의 물리적 경로 얻어오기

예제 2-10] WEB-INF/web.xml

```
<context-param>
    <param-name>INIT_PARAM</param-name>
    <param-value>web.xml에 저장한 초기화 매개변수</param-value>
</context-param>
```

예제 2-10] 02ImplicitObject/ApplicationMain.jsp

```
<body>
    <h2>web.xml에 설정한 내용 읽어오기</h2>
    초기화 매개변수 : <%= application.getInitParameter("INIT_PARAM") %> ❶

    <h2>서버의 물리적 경로 얻어오기</h2>
    application 내장 객체 : <%= application.getRealPath("/02ImplicitObject") %> ❷
```

❷ /02ImplicitObject 폴더의 물리적 경로를 얻어옴. 이클립스에서는 .metadata 하위로 설정됨

2.5 application 객체

▣ application 객체

예제 2-10] 02ImplicitObject/ApplicationMain.jsp

```
<h2>선언부에서 application 내장 객체 사용하기</h2>
<%!
public String useImplicitObject() {  ❸
    return this.getServletContext().getRealPath("/02ImplicitObject");
}
public String useImplicitObject(ServletContext app) {  ❹
    return app.getRealPath("/02ImplicitObject");
}
%>
<ul>
    <li>this 사용 : <%= useImplicitObject() %></li>  ❺
    <li>내장 객체를 인수로 전달 : <%= useImplicitObject(application) %></li> ❻
</ul>
```

❸ getServletContext() 를 통해 application 내장객체를 얻음
❹ 매개변수를 통해 application 내장객체를 얻음

2.6 exception 객체

▣ exception 객체

- 오류명과 오류 메시지를 출력하는 부분에서 사용
- JSP에서 그 이상으로 사용되는 경우가 거의 없으므로 오류페이지 처리를 위한 또 다른 방식을 학습
- 웹 프로그래밍에서 주로 발생되는 에러

에러 코드	에러 메시지	조치 방법
404	Not Found	요청한 경로에서 문서를 찾을 수 없음 경로명이나 파일명이 제대로 입력되었는지 확인
405	Method Not Allowed	허용되지 않는 전송방식 get 혹은 post 요청시 이를 처리할 컨트롤러가 있는지 확인
500	Internal Server Error	서버 내부 오류 가장 많이 발생하는 에러 오타나 로직의 오류가 있는지 개발 중인 코드를 전반적으로 확인해야 함

2.6 exception 객체

▣ exception 객체

예제 2-12] WEB-INF/web.xml

```
<error-page>
    <error-code>404</error-code>  ❶ 에러 코드
    <location>/02ImplicitObject/Exception.jsp</location>  ❷ 출력할 페이지
</error-page>
```

404에러가 발생하면 Exception.jsp에서 에러를 처리하는 형식으로 텍스트, 이미지를 적절히 출력함

예제 2-13] 02ImplicitObject/Exception.jsp

```
<%
int status = response.getStatus();  // response 내장 객체로부터 에러 코드 확인

// 에러 코드에 따라 적절한 메시지 출력
if (status == 404) {
    out.print("404 에러가 발생하였습니다.");
    out.print("<br/>파일 경로를 확인해주세요.");
}
```

학습 마무리

- JSP의 내장 객체는 별도의 선언 없이 사용할 수 있음
- 클라이언트의 요청을 받거나 요청에 대한 응답을 할 수 있음
- 자바 코드에 대한 예외 처리를 할 수 있음

▣ 핵심요약

- request 객체 : 클라이언트의 요청을 받거나 웹 브라우저에 대한 정보 혹은 요청 헤더에 대한 정보를 읽을 때 사용
- response 객체 : 요청에 대한 응답을 웹 브라우저로 보낼 때 사용. 페이지 이동이나 응답 헤더를 추가할 때도 사용.
- out 객체 : 변수 등의 값을 웹 브라우저에 출력할 때 주로 사용.
- application 객체 : 웹 애플리케이션을 구성하는 모든 JSP에서 접근 가능한 객체로, 웹 애플리케이션에 대한 설정값을 저장할 때 주로 사용.
- exception 객체 : 예외 처리를 위해 사용.