3강 자료

기본 객체

기본 객체	데이터 타입	설명
request	javax.servlet.http.HttpServletRequest	클라이언트의 요청 정보
response	javax.servlet.http.HttpServletResponse	응답 정보
pageContext	javax.servlet.jsp.PageContext	jsp 페이지에 대한 정보
session	javax.servlet.jsp.HttpSession	http 세션 정보
application	javax.servlet.ServletContext	웹 어플리케이션에 대한 정보
out	javax.servlet.jsp.JspWriter	jsp 페이지가 생성하는 결과를 출력하는 출력 스트림
config	javax.servlet.ServletConfig	jsp 페이지에 대한 설정 정보
page	java.lang.Object	jsp 페이지를 구현하는 자바 클래스 인스턴스
exeption	java.lang.Throwable	예외(Exception) 객체, 예외처리 페이지에서 사용

기본 객체 Out

- JSP가 생성하는 모든 내용은 out 객체를 통해 전송
  - HTML 태그, 텍스트, 표현식
  - 출력 가능 형식: 기본 데이터 타입(Boolean, char, char[], double, float, int, long)과 String

예 <HTML> -> <% out.println("<HTML>"); %>

```
<%
   if(grade > 10) {
%>
   <%= "10보다 크다. %>
<%
   } else {
 %>
   <%= "10이하 이다. %>
<%
   }
%>
```

```
<%
   if(grade > 10) {
      out.println("10보다 크다." );
   else {
      out.println("10이하 이다.");
 }
%>
```

기본 객체 Out

- out 객체의 버퍼 관련 메서드

메서드	리턴 타입	설명
getBufferSize()	int	버퍼의 크기를 반환
getRemaining()	int	현재 버퍼의 남은 크기를 반환
clear()	void	버퍼의 내용을 비운다. Flush한 뒤에는 IOException 발생
clearBuffer()	void	버퍼의 내용을 비운다. Flush한 뒤에도 IOException 발생하지 않는다.
flush()	void	버퍼의 내용을 클라이언트에 전송
isAutoFlush()	boolean	버퍼가 찼을 때 자동 flush 여부를 확인

기본 객체 Out

- out 객체의 버퍼 관련 메서드 사용 예
- [리스트 5.2]

```
<%
   버퍼크기: <%= out.getBufferSize() %>
   남은 버퍼크기: <%= out.getRemaining () %>
   auto flush 여부: <%= out.isAutoFlush() %>
%>
```

기본 객체 pageContext

- JSP 페이지와 일대일로 연결된 객체
  - 기본 객체 구하기
  - 속성 처리하기
  - 페이지의 흐름 제어
  - 에러 데이터 구하기

기본 객체 pageContext

- pageContext 객체의 메서드: 기본 객체 구하기

메서드	리턴 타입	설명
getRequest()	ServletRequest	reqeust 객체
getResponse()	ServletResponse	response 객체
getSession()	HttpSession	session 객체
getServletContext()	ServletContext	application 객체
getServletConfig()	ServletConfig	config 객체
getOut()	JspWriter	out 객체
getException()	Exception	Exception 객체 – exception 페이지에서만 사용 가능
getPage()	Object	page 객체

기본 객체 pageContext

- [리스트 5.3]

```
<%
   HttpSrevletRequest httpRequest =
      (HttpServletRequest)pageContext.getRequest();
%>
   request 기본 객체와 pageContext.getRequest()의 동일 여부:
<%
   <%= request == httpRequest %>
   <br />
   pageContext.getOut() 메서드를 사용한 데이터 출력
<%
   pageContext.getOut().println(“Hello”);
%>
```

기본 객체 application

- 모든 JSP 페이지는 하나의 application 객체를 공유
  - application의 초기 파라미터
    - WEB-INF/web.xml
    - 웹 어플리케이션을 위한 초기 파라미터를 포함하는 설정 내용
- application 객체 관련 메서드

메서드	리턴 타입	설명
getInitParameter(String name)	String	이름이 name인 초기화 파리미터를 읽어 반환 없으면 null
getParameterNames()	Enumeration<String>	초기화 파라미터의 이름 목록을 열거형으로 반환

기본 객체 application

- [리스트 5.4] web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
      http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
      version="3.1">
  <context-param>
      <description>로깅 여부</description>
      <param-name>logEnabled</param-name>
      <param-value>true</param-value>
  </context-param>
  <context-param>
      <description>디버깅 레벨</description>
      <param-name>debugLevel</param-name>
      <param-value>5</param-value>
  </context-param>
</web-app>
```

기본 객체 application

- [리스트 5.5] readInitParameter.jsp

```
<%
    Enumeration<String> initParamEnum =
        application.getInitParameterNames();
    while (initParamEnum.hasMoreElements()) {
        String initParamName = initParamEnum.nextElement();
%>
<li><%= initParamName %> =
<%= application.getInitParameter(initParamName) %>
<%
    }
%>
```

기본 객체 application

- 초기화 파라미터는
  - 데이터베이스 연결 정보 등
  - 어플리케이션을 실행하는 데 필요한 주요 정보

- 서버 정보 확인 메서드

메서드	리턴 타입	설명
getServerInfo()	String	서버 정보
getMajorVersion()	String	서버가 지원하는 서블릿의 major 버전
getMinorVersion()	String	서버가 지원하는 서블릿의 minor 버전

기본 객체 application

- [리스트 5.6] viewServerInfo.jsp

```
<%
  서버 정보: <%= application.getServerInfo() %><br />
 서블릿 major 버전: <%= application.getMajorVersion() %><br />
 서블릿 minor 버전: <%= application.getMinorVersion() %>
%>
```

기본 객체 application

- Application를 이용하여 웹 컨테이너가 사용하는 로그 파일에 로그 메시지 기록

메서드	리턴 타입	설명
log(String msg)	void	msg 문자열을 로그 파일에 기록
log(String msg, Throwable throwable)	void	mMsg 문자열을 로그 파일에 기록 Exception 정보를 함께 기록

- [리스트 5.7][리스트 5.8] useApplicaionLog.jsp, useJspLog.jsp

```
<%
    로그 파일 확인1 application.log("로그 메시지1");
    로그 파일 확인2 log("로그 메시지2");
%>
```

기본 객체 application

- Application를 이용하여 자원 접근

메서드	리턴 타입	설명
getRealPath(String path)	String	path에 주어진 자원의 실제 경로를 반환
getResource(String path)	java.net.url	path에 주어진 자원의 url을 반환
getResourceAsStream (String path)	java.io.InputStream	Path에 주어진 자원을 입력 받을 수 있는 InputStream을 반환

- [리스트 5.11], [리스트 5.12]

기본 객체 application

```
<%@ page import="java.io.*" %>
<%@ page import="java.net.*" %>
```

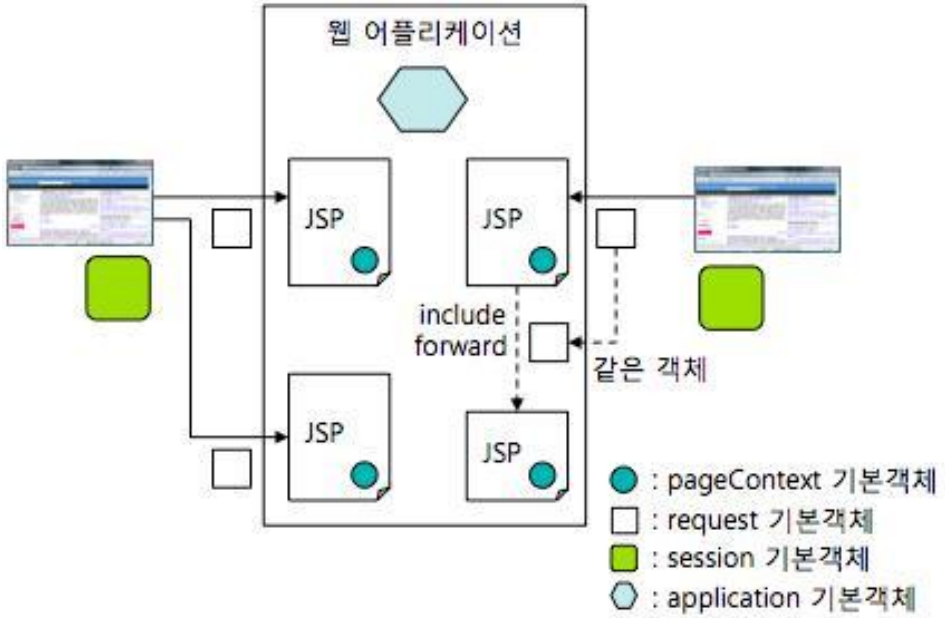
```
<%
    String fileName = "/message/notice.txt";
    URL urlResource = new URL(fileName);
%>
자원의 실제 경로 확인:
<%= application.getRealPath(fileName) %>
<%
    URL urlObject =
         application.getResource(fileName);
%>
URL 객체의 정보 확인: <br />
호스트 정보 => <%= urlObject.getHost() %><br />
포트 정보 => <%= urlObject.getPort() %><br />
%>
```

기본 객체 application

```
AS Stream으로 읽기<br>
<%
    char[] buff = new char[128];
    int len = -1;
    try {
        InputStreamReader ir =
           new InputStreamReader(
               application.getResourceAsStream(fileName),
               "UTF-8");
        while( (len = ir.read(buff)) != -1 ) {
            out.print(new String(buff, 0, len));
        }
    }catch(IOException ex) {
         out.println("Exception 발생: " + ex.getMessage());
    }
%>
```

기본 객체 사용 영역

- Page 영역
  - 하나의 JSP 페이지를 처리할 때 사용되는 영역
  - pageContext 객체
- Request 영역
  - 하나의 HTTP 요청을 처리할 때 사용되는 영역
  - request 객체
- Session 영역
  - 하나의 웹 브라우저와 관련된 영역
  - session 객체
- Application 영역
  - 하나의 웹 어플리케이션과 관련된 영역
  - application 객체



기본 객체 attribute 사용하기

- attribute는 데이터 공유 목적

메서드	리턴 타입	설명
setAttribute(String name, Object value)	void	name으로 value 저장
getAttribute(String name)	Object	name의 값 읽기
removeAttribute(String name)	void	name attribute 삭제
getAttributeNames()	Enumeration<String>	attribute의 name 목록 가져오기 (pageContext는 지원X)

- [리스트 5.13]

기본 객체 attribute 활용

기본 객체	영역	용도
pageContext	Page	하나의 JSP 페이지 내에서 공유할 값을 저장 커스텀 태그에서 새로운 변수를 추가할 때 사용
request	Request	한 번의 요청을 처리하는 데 사용되는 모든 JSP 페이지에서 공유할 값을 저장 하나의 요청을 처리하는 데 사용되는 JSP 페이지 간의 정보 전달 목적으로 사용
session	Session	한 사용자와 관련된 JSP 페이지에서 공유할 값을 저장 한 사용자의 요청을 처리하는 데 사용되는 JSP 페이지 간의 정보 전달 목적으로 사용
application	Application	모든 사용자가 공유할 값을 저장 임시 폴더 경로와 같은 웹 어플리케이션의 설정 정보 저장

에러처리: Exception 직접 처리하기

- request 파라미터가 없는 상태에서 발생하는 오류
- [리스트 6.1] 참조

```
<%
    name 파라미터 <%= request.getParameter("name").toUppereCase() %>
%>
```

에러처리를 하지 않으면,
tomcat이 에러처리

문제점:
- 시스템의 에러 화면으로 신뢰도 저하
- 코드 노출로 인한 보안 문제

HTTP 상태 500 – 내부 서버 오류

타입 예외 보고

메시지 행 [10]에서 [/list0601.jsp]을(를) 처리하는 중 예외 발생

설명 서버가, 해당 요청을 충족시키지 못하게 하는 예기치 않은 조건을 맞닥뜨렸습니다.

예외

```
org.apache.jasper.JasperException: 행 [10]에서 [/list0601.jsp]을(를) 처리하는 중 예외 발생

7: <title>Insert title here</title>
8: </head>
9: <body>
10: name = <%= request.getParameter( "name").toUpperCase() %>
11: </body>
12: </html>

Stacktrace:
        org.apache.jasper.servlet.JspServletWrapper.handleJspException(JspServletWrapper.java:599)
        org.apache.jasper.servlet.JspServletWrapper.service(JspServletWrapper.java:488)
        org.apache.jasper.servlet.JspServlet.serviceJspFile(JspServlet.java:380)
        org.apache.jasper.servlet.JspServlet.service(JspServlet.java:328)
        jakarta.servlet.http.HttpServlet.service(HttpServlet.java:631)
        org.apache.tomcat.websocket.server.WsFilter.doFilter(WsFilter.java:53)

근본 원인 (root cause)

java.lang.NullPointerException: Cannot invoke "String.toUpperCase()" because the return value of "jakarta.servlet.http.HttpServletRequest.getParameter(String)" is null
        org.apache.jsp.list0601_jsp._jspService(list0601_jsp.java:130)
        org.apache.jasper.runtime.HttpJspBase.service(HttpJspBase.java:70)
        jakarta.servlet.http.HttpServlet.service(HttpServlet.java:631)
        org.apache.jasper.servlet.JspServletWrapper.service(JspServletWrapper.java:455)
        org.apache.jasper.servlet.JspServlet.serviceJspFile(JspServlet.java:380)
        org.apache.jasper.servlet.JspServlet.service(JspServlet.java:328)
        jakarta.servlet.http.HttpServlet.service(HttpServlet.java:631)
        org.apache.tomcat.websocket.server.WsFilter.doFilter(WsFilter.java:53)
```

비고 근본 원인(root cause)의 풀 스택 트레이스를, 서버 로그들에서 확인할 수 있습니다.

기본 객체 attribute 활용

- try~catch 문을 사용하여 코드 내에서 직접 오류 처리
- [리스트 6.2] 참조

```
<%
    try {
%>
<%
        name 파라미터 <%= request.getParameter("name").toUppereCase() %>
%>
<%
    catch(Exception ex) {
%>
        name 파라미터가 올바르지 않습니다.
<%
    }
%>
```

에러처리: Exception 직접 처리하기

- try~catch 문을 사용하여 코드 내에서 직접 오류 처리
- [리스트 6.2] 참조

```
<%
    try {
%>
<%
        name 파라미터 <%= request.getParameter("name").toUppereCase() %>
%>
<%
    catch(Exception ex) {
%>
        name 파라미터가 올바르지 않습니다.
<%
    }
%>
```

에러처리: Exception 직접 처리하기

- try~catch 문을 사용하여 코드 내에서 직접 오류 처리
- [리스트 6.3] 참고

```
<%@ page errorPage="/error/viewErrorMessage.jsp" %>

<%
    name 파라미터 <%= request.getParameter("name").toUppereCase() %>
%>
```

- [리스트 6.4] 참조

```
<%@ page isErrorPage="true" %>
<body>
요청을 처리하는 과정에서 에러가 발생했습니다.<br />
<br />
에러 타입: <%= exception.getClass().getName() %><br />
에러 내용: <%= exception.getMessage() %><br />
</body>
```

에러처리: 응답 코드별 에러 페이지 처리

- web.xml 파일에서 에러 코드와 에러 코드별 페이지 지정

```
<error-page>
   <error-code>404</error-code>
   <location>/error/error404.jsp</location>
</error-page>
<error-page>
   <error-code>500</error-code>
   <location>/error/error500.jsp</location>
</error-page>
```

에러처리: 응답 코드별 에러 페이지 처리

- [리스트 6.6] – error404.jsp 참고

에러처리: 응답 코드별 에러 페이지 처리

```
<title>404 - Error Page</title>
</head>
<body>
<br />
<center><p style="font-size:30pt;">404 Not Found</p><br />
요청한 페이지는 존재하지 않습니다.
</center>
</body>
```

http 주요 에러 코드

- 상태 코드: 서버에서 사용자에게 알려준다.
- https://ko.wikipedia.org/wiki/HTTP_상태코드

상태코드	설명
200	요청을 정상 처리
307	임시로 페이지를 리다이렉트 함. response.sendRedirect() 이용할 때
400	잘못된 구문의 사용자 요청(서버가 사용자의 요청을 해석하지 못함)
403	요청을 거부함(Access Denied)
404	File Not Fount(자원이 존재하지 않음)
405	요청한 메서드(GET, POST 등)를 허용하지 않음
500	서부 내부 에러 발생(JSP 페이지에서 Exception 발생)
503	서버가 일시적으로 서비스를 제공할 수 없음

에러처리: Exception별 에러 페이지 처리

- web.xml 파일에서 Exception 타입별 에러 페이지 지정

```
<error-page>
    <exception-type>java.lang.NullPointerException</error-code>
    <location>/error/errorNullPointerException.jsp</location>
</error-page>
<error-page>
    <exception-type>org.apache.jasper.JasperException</error-code>
    <location>/error/errorJasperException.jsp</location>
</error-page>
```

에러 페이지 우선순위

- 1. page 디렉티브의 errorPage 속성에 지정한 페이지
- 2. web.xml 파일의 <exception-type>과 <location>에서 지정한 페이지
- 3. web.xml 파일의 <error-code>와 <location>에서 지정한 페이지
- 4. 웹 컨테이너가 제공하는 기본 에러 페이지

에러 페이지 우선순위

- 에러가 발생하기 전에 버퍼 플러시가 동작하면,
  - 버퍼의 내용이 사용자에게 전송되고
  - 에러 메시지가 그 뒤에 붙는다.
- [리스트 6.8] 참고

<%@ page
errorPage="/error/viewErrorMessage.jsp" %>를 빼면,
web.xml에 정의된 Exception 처리 페이지로 바로 이동
web.xml이 없으면, 내용은 정상 출력되고, 에러 처리는
Eclipse의 Console 창에 표시된다.

```
<%@ page buffer="1kb" %>
<%@ page errorPage="/error/viewErrorMessage.jsp" %>
…
<body>
버퍼 크기는 <%= pageContext.getOut().getBufferSize() %> 이다.<br /><br />
현재 autoFlush <%= out.isAutoFlush() %><br />
<% for(int i = 0; i < 300; i++) { out.print(i); } %>
<%= 1 / 0 %>
</body>
```

End of Document

Thank you