

# Java Web Programming 입문

(Servlet/JSP)

25일차

# 오늘의 키워드

- ▶ 내장 객체 (Implicit Object)
  - out
  - application
  - config
  - page
  - exception

# out

## ▶ out 내장 객체

### ◦ 는...

- Web Browser에 Data를 전송하기 위한 출력 Stream
- JSPWriter Class의 Instance
- pageContext Object의 getOut() Method로 자동 생성

```
JspWriter out = null; out = pageContext.getOut();
```

### ◦ 기능

- Web Browser에 출력될 내용을 지정
- JSP Page의 출력과 관련된 Buffer를 관리

# out

## ▶ out 내장 객체

메소드 원형	메소드의 기능
void print()	데이터를 출력한다.
void println()	데이터를 출력한 후 줄 바꿈을 한다.
void newLine()	줄 바꿈을 한다.

## ▶ printTest.jsp

```
<%@ page contentType = "text/html; charset=euc-kr" %>

<html>
  <head>
    <title>out 객체의 print</title>
  </head>
  <body>
    <%
      out.print("test1 ");
      out.println("test2");
      out.print("test3");
      out.newLine();
      out.print("test4");
    %>
  </body>
</html>
```

# out

## ▶ Page 지시문의 Buffer

### ◦ Page 지시문

- JSP Page를 실행하는데 필요한 정보를 지정
  - 문서 타입
  - Class Import 정보
  - Error Page
- 속성
  - buffer
  - autoFlush

# out

## ▶ page 지시문의 buffer 속성

- 출력될 Buffer 크기 지정 `<%@ page buffer="1kb"%>`
  - out 내장 객체를 이용하여 Data를 Stream에 기록할 때
    - 기록된 Data의 크기가 1Kb가 되면
      - 자동으로 Client에 전송
    - 다시 Buffer가 채워지면 Data 전송

## ▶ page 지시문의 autoFlush 속성

- 자동 Flush 여부를 지정 `<%@ page autoFlush="false"%>`

## ▶ pageContext 내장 객체의

- 생성 코드
- Default
  - 8192byte, true

```
pageContext =  
    _jspxFactory.getPageContext (  
        this,  
        request,  
        response,  
        null,  
        true,  
        1024,  
        false  
    );
```

# out

## ▶ out 내장 객체의 Buffer 관련

메소드 원형	메소드의 기능
int getBufferSize()	JspWriter에 의해 사용 중인 버퍼의 크기를 반환
int getRemaining()	버퍼의 남은 공간을 반환
void flush()	출력 스트림에 버퍼링 되어 남아 있는 데이터를 실제 스트림으로 보낸다.
boolean isAutoFlush()	자동 Flush 여부를 반환한다.
void clear()	버퍼의 내용을 지운다. 버퍼가 이미 Flush 되어 있다면 IOException을 발생시킨다.
void clearBuffer()	버퍼의 현재 내용을 지운다. 버퍼가 이미 Flush 되어 있더라도 IOException을 발생시키지 않는다.

```
<%@ page buffer="1kb" autoFlush="false" %>
```

```
out.print("<h3>버퍼의 크기 : " + out.getBufferSize() + "</h3>");  
out.print("<h3>자동 Flush 여부 : " + out.isAutoFlush() + "</h3>");
```

```
out.print("<h3>남은 버퍼의 크기 : " + out.getRemaining() + "</h3>");  
out.flush();  
out.print("<h3>남은 버퍼의 크기 : " + out.getRemaining() + "</h3>");
```

# out

## ▶ bufferTest.jsp

```
<%@ page buffer="1kb" autoFlush="false" %>

<%@ page contentType="text/html; charset=euc-kr" %>
<%@ page buffer="1kb" autoFlush="false" %>
<html>
  <head>
    <title>out 객체의 버퍼 관리</title>
  </head>
  <body>
    <%
      out.print("<h3>버퍼의 크기 : " + out.getBufferSize() + "</h3>");
      out.print("<h3>자동 Flush 여부 : " + out.isAutoFlush() + "</h3>");
      out.print("<h3>남은 버퍼의 크기 : " + out.getRemaining() + "</h3>");
      out.flush();
      out.print("<h3>남은 버퍼의 크기 : " + out.getRemaining() + "</h3>");
    %>
  </body>
</html>
```



# application

## ▶ application 내장 객체

### ◦ 는...

- javax.servlet.ServletContext Interface의 Instance
- 하나의 Web Application 관리
- Web Application 내 정보 공유

### ◦ 생성

```
ServletContext application = null;  
application = pageContext.getServletContext();
```

### ◦ ServletContext Object

- 하나의 Web Application에 하나의 ServletContext
- Web Application 내 모든 Servlet 관리 및 정보 공유
  - Web Application의 등록정보?

# application

## ▶ application 내장 객체

### ◦ 기능

- Server 정보 출력
  - Servlet 정보
  - Web Application의 전반적인 정보
  - Servlet Engine 정보
  - etc...
- Web Application의 초기화 Parameter 사용
  - web.xml 파일을 이용
    - 초기화 정보 제공 및 공유
- Application 내 정보 저장 및 공유
  - ServletContext에 Object 등의 정보 저장 및 공유
  - 타 Servlet Data 공유
  - Log 기록
- PageContext Object 처럼
  - Page의 제어권을 넘기기 (forward())
  - 처리 후 다시 제어권 돌려받기 (include())

# application

- ▶ application 내장 객체로 log 작성

```
application.log("::: Print Server Information using application Object");
```

- ▶ 그 외 application Method

메소드 원형	메소드의 기능
String getMimeType(String file)	지정된 파일의 MIME 타입을 출력
String getServerInfo()	서버 프로그램의 정보 출력
String getRealPath(String path)	지정한 자원의 실제 시스템 경로를 출력
int getMajorVersion()	서버가 지원하는 서블릿의 메이저 버전을 출력
int getMinorVersion()	서버가 지원하는 서블릿의 마이너 버전을 출력

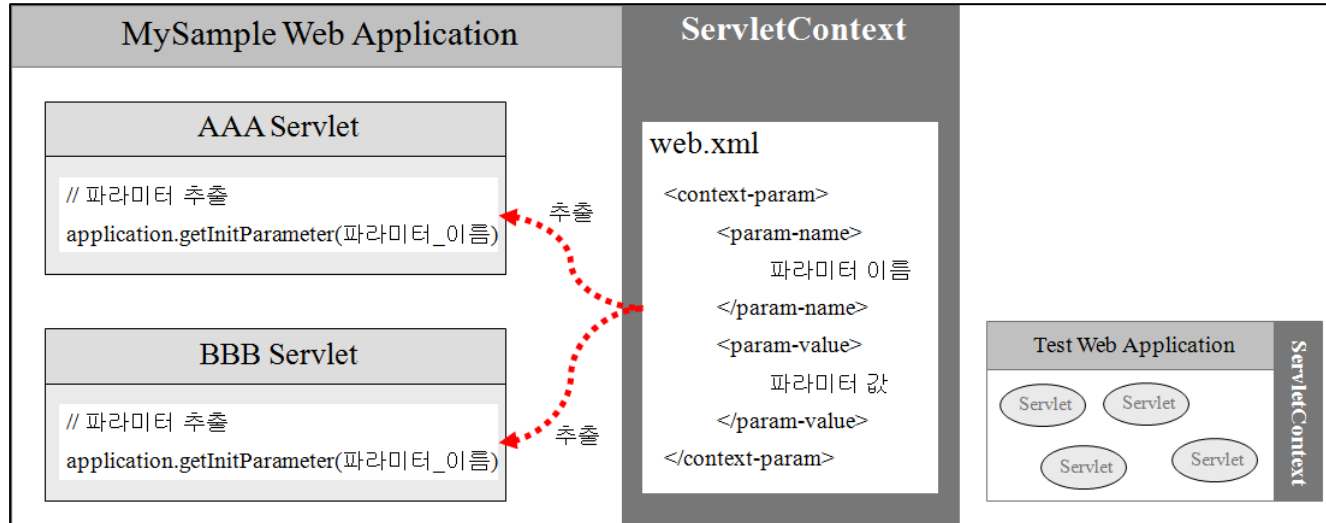
# 내장 객체 (Implicit Object)

## ▶ printServerInfo.jsp

```
<%@ page contentType = "text/html; charset=euc-kr" %>
<html>
  <head>
    <title>서버 정보 출력</title>
  </head>
  <body>
    <%
      application.log("::: Print Server Information using application Object");
    %>
    MimeType -
    <%=application.getMimeType("jabook.gif")%><br>
    ServerInfo -
    <%=application.getServerInfo()%><br>
    RealPath -
    <%=application.getRealPath("printServerInfo.jsp")%><br>
    MajorVersion -
    <%=application.getMajorVersion()%><br>
    MinorVersion -
    <%=application.getMinorVersion()%><br>
  </body>
</html>
```

# 내장 객체 (Implicit Object)

## ▶ Web Application Parameter 공유



- Web Application에 등록된 초기화 Parameter 이용
- Web Application에는
  - Web Application 관련 정보를 담고 있는 web.xml 파일이 하나씩 존재
  - web.xml 파일에 Web Application의 초기화 Parameter 등록할 수 있음
  - web.xml 파일에 등록된 초기화 Parameter를
    - application 내장 객체를 이용하여 사용 가능
  - web.xml 파일에 등록된 초기화 Parameter는
    - 동일한 Web Application에 존재하는 모든 Servlet이 공유 할 수 있음

# application

## ▶ Parameter 등록

- web.xml 파일에 초기화 Parameter 정보를 작성
  - web.xml
    - Web Application의 환경 설정 정보를 저장
    - Web Application의 WEB-INF 아래 존재
  - web.xml 파일에 Parameter 작성하는 형식

```
<context-param>  
  <param-name>파라미터 이름</param-name>  
  <param-value>파라미터 값</param-value>  
</context-param>
```

- <context-param> Tag
  - <web-app> Tag의 하위 요소
  - <param-name>, <param-value> Tag를 이용
    - Parameter의 이름, 값을 기술

```
<context-param>  
  <param-name>debugMessage</param-name>  
  <param-value>1</param-value>  
</context-param>
```

# application

## ▶ web\_MySample\_00.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
  <web-app xmlns="http://java.sun.com/xml/ns/j2ee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
      http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd"
    version="2.4">
    <display-name>Welcome to Tomcat</display-name>
    <description>      Welcome to Tomcat      </description>
    <servlet>
      <servlet-name>org.apache.jsp.index_jsp</servlet-name>
      <servlet-class>org.apache.jsp.index_jsp</servlet-class>
    </servlet>
    <servlet-mapping>
      <servlet-name>org.apache.jsp.index_jsp</servlet-name>
      <url-pattern>/index.jsp</url-pattern>
    </servlet-mapping>
    <context-param>
      <param-name>debugMessage</param-name>
      <param-value>1</param-value>
    </context-param>
    <context-param>
      <param-name>executionLevel</param-name>
      <param-value>5</param-value>
    </context-param>
  </web-app>
```

# application

- ▶ application 내장 객체의 초기화 Parameter 관련 Method

메소드 원형	메소드의 기능
Enumeration getInitParameterNames()	파라미터의 이름 집합을 반환
String getInitParameter(String name)	이름이 name인 파라미터 값을 반환

- ▶ usingParam.jsp

```
<%@ page contentType = "text/html; charset=euc-kr" %>
<%@ page import="java.util.*"%>
<html>
  <head>
    <title>파라미터 사용</title>
  </head>
  <body>
    <%
      Enumeration paramEnum = application.getInitParameterNames();
      while(paramEnum.hasMoreElements()) {
        String paramName = (String)paramEnum.nextElement();
        String paramValue = application.getInitParameter(paramName);
        out.println(paramName + " - " + paramValue + "<br>");
      }
    %>
  </body>
</html>
```



# application

## ▶ application VS pageContext

- 정보 공유 측면에서의 공통점/차이점
  - pageContext 내장 객체
    - 하나의 Page당 하나의 pageContext 내장 객체 존재
    - 하나의 Page(Servlet) 내에서만 정보 공유 가능
  - application 내장 객체
    - 하나의 Web Application 당 하나의 application 내장 객체 존재
    - Web Application 내에서 정보 공유

# config

## ▶ config 내장 객체

### ◦ 는...

- javax.servlet.ServletConfig Interface의 Instance
- Servlet Container를 관리
- Servlet Container에 존재하는 모든 Web Application들에게
  - Servlet을 초기화하는 동안 참조해야 할 정보를 공유
  - Servlet 초기화 시 참조해야 할 여러 정보를 가지고 있다가 사용

### ◦ 생성

```
ServletConfig config = null; config = pageContext.getServletConfig();
```

### ◦ 기능

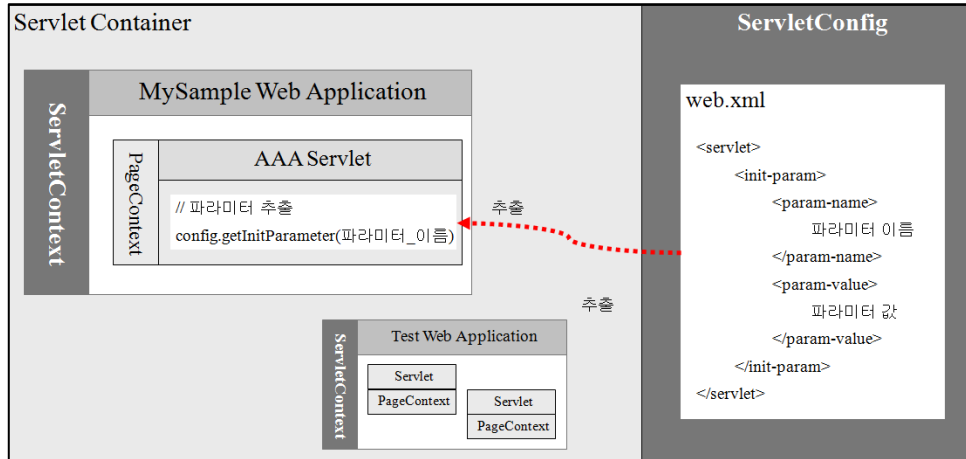
- Servlet에 대한 정보 출력
- Servlet Container의 초기화 Parameter 사용
- 모든 Web Application이 공유하는 초기화 Parameter를 사용

```
ServletContext servletContext = config.getServletContext();
```

- 필요없다. (application 내장 객체 참조)

# config

## ▶ Servlet Container 초기화 Parameter



- config 내장 객체는
  - 모든 Web Application을 포함하는
    - Servlet Container를 관리
- config 내장 객체를 이용해서
  - Servlet Container에 등록된 초기화 Parameter 읽기/사용 가능
- Servlet Container에 존재하는 web.xml 파일
  - Servlet Container에 속한 모든 Web Application에 적용
  - 이 web.xml 파일에 등록된 초기화 Parameter는 config 내장 객체를 이용하여 공유

# config

- ▶ 전역 초기화 Parameter
  - WEB-INF에 있는 web.xml 이 아니야!
  - .../conf/ 에 web.xml
    - Servlet Container에 대한 환경 설정 파일
    - Servlet Container에 존재하는 모든 Web Application이 공유
  - 작성 형식

```
<servlet>
  <servlet-name> 서블릿 이름 </servlet-name>
  <servlet-class>org.apache.jasper.servlet.JspServlet</servlet-class>
  <init-param>
    <param-name> 파라미터 이름 </param-name>
    <param-value> 파라미터 값 </param-value>
  </init-param>
</servlet>
```

# config

## ▶ web\_ServletContainer.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
    http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd"
  version="2.4">
  <!-- 생략 -->
  <servlet>
    <servlet-name>jsp</servlet-name>
    <servlet-class>org.apache.jasper.servlet.JspServlet</servlet-class>
    <init-param>
      <param-name>fork</param-name>
      <param-value>>false</param-value>
    </init-param>
    <init-param>
      <param-name>xpoweredBy</param-name>
      <param-value>>false</param-value>
    </init-param>
    <load-on-startup>3</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>jsp</servlet-name>
    <url-pattern>*.jsp</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>jsp</servlet-name>
    <url-pattern>*.jspx</url-pattern>
  </servlet-mapping>
  <!-- 생략 -->
</web-app>
```

# config

## ▶ 전역 초기화 Parameter

메소드 원형	메소드의 기능
String getServletName()	서블릿의 이름을 반환
Enumeration getInitParameterNames()	파라미터의 이름 집합을 반환
String getInitParameter(String name)	이름이 name인 파라미터 값을 반환

## ▶ useConfig.jsp

```
<%@ page contentType="text/html; charset=euc-kr" %>
<%@ page import="java.util.Enumeration" %>
<html>
  <head>
    <title>confign 내장 객체</title>
  </head>
  <body>
    <%
      out.println("<h3>Servlet Name</h3>" + config.getServletName() + "<br>");
      out.println("<h3>Parameter Information</h3>");
      Enumeration paramNames = config.getInitParameterNames();
      while(paramNames.hasMoreElements()) {
        String paramName = (String)paramNames.nextElement();
        String paramValue = config.getInitParameter(paramName);
        out.println(paramName + " - " + paramValue + "<br>");
      }
    %>
  </body>
</html>
```

# page

## ▶ page 내장 객체

### ◦ 는...

- JSP Page로부터 생성되는 Servlet Object를 참조
  - 참조 변수
  - JSP Page 자신이 생성할 Servlet Object를 참조하는 참조 객체

### ◦ 생성

- `_jspService()` Method의 내부 변수로 선언

```
Object page = this;
```

### ◦ 기능

- 자주 사용하지 않는다(?)
- 주로 내부적인 작업용
- 외부에서 사용할 땐 형 변환 후 (Object니까)
  - 멤버 변수나, 멤버 메소드 지정 용도

# page

## ▶ page 내장 객체

### ◦ 사용

#### • 선언문

```
<%!  
    int sum(int a, int b) {  
        return a + b;  
    }  
%>
```

#### • 멤버 메소드의 사용

```
int sum1 = sum(aaa, bbb);
```

#### • this 를 이용한 멤버 메소드의 사용

```
int sum2 = this.sum(aaa, bbb);
```

#### • page 내장 객체를 이용한 멤버 메소드 사용

```
int sum3 = ((pageObject_jsp)page).sum(aaa, bbb);
```



# page

## ▶ pageObject.jsp

```
<%@ page contentType="text/html; charset=euc-kr" %>

<%!
    int sum(int a, int b) {
        return a + b;
    }
%>
<html>
    <head>
        <title>page 내장 객체</title>
    </head>
    <body>
<%
    int aaa = 4, bbb = 5;
    out.println("처음 생성된 객체를 이용해서 출력 : " + sum(aaa, bbb) + "<br>");
    out.println("this 객체를 이용해서 출력 : " + this.sum(aaa, bbb) + "<br>");
    out.println("page 내장 객체를 이용해서 출력 : "
        + ((pageObject_jsp)page).sum(aaa, bbb) + "<br>");
%>
    </body>
</html>
```

# exception

## ▶ exception 내장 객체

- 는...
  - java.lang.Throwable Class 형의 JSP 내장 객체
  - 에러 처리를 위한 객체
  - Page 실행 시
    - Servlet이 처리하지 못한 에러가 발생하거나,
    - 예외 처리 Page를 지정하였을 경우
    - 지정된 Page로 예외를 전달하는 역할
- 기능
  - JSP 처리 도중 발생하는 에러 정보 출력
- 사용
  - page 선언문에서 isErrorPage 속성 값 “true”로 설정
    - 이 속성을 정의하지 않으면 사용 불가

```
<%@ page isErrorPage="true"%>
```

- 에러를 처리할 JSP Page 지정

```
<%@ page errorPage="calculErr.jsp"%>
```

- buffer 속성을 “false” 로 설정하면 안됨

# exception

## ▶ calcul\_exception.jsp

```
<%@ page contentType="text/html; charset=euc-kr" %>

<html>
  <head>
    <title>에러 페이지 지정</title>
  </head>
  <body>
    <%=100/0%>
  </body>
</html>
```

```
<%@ page  errorPage="에러 처리 페이지" %>
```

```
<%@ page errorPage="calculErr.jsp" %>
```

```
<%@ page  isErrorPage="true | false" %>
```

```
<%@ page isErrorPage="true" %>
```

```
에러 형식 : <%=exception.getClass().getName()%>
에러 메시지 : <%=exception.getMessage()%>
```

# exception

## ▶ calcul.jsp

```
<%@ page contentType="text/html; charset=euc-kr" %>
<%@ page errorPage="calculErr.jsp" %>

<html>
  <head>
    <title>에러 페이지 지정</title>
  </head>
  <body>
    <%=100/0%>
  </body>
</html>
```

## ▶ calculErr.jsp

```
<%@ page contentType="text/html; charset=euc-kr" %>
<%@ page isErrorPage="true" %>

<html>
  <head>
    <title>에러 처리 페이지</title>
  </head>
  <body>
    <h3>에러 처리 페이지</h3>
    에러 형식 : <%=exception.getClass().getName()%><br>
    에러 메시지 : <%=exception.getMessage()%>
  </body>
</html>
```

# exception

## ▶ 에러 코드별 에러 처리

### ◦ web.xml 이용

- Tomcat에서는 사용자가 특정 에러 코드에 따라
  - 출력될 Web Page 지정가능
  - 타 Server 제품군은 Case By Case (비스무리하게 있다)
- 작성 형식

```
<error-page>  
    <error-code>에러 코드</error-code>  
    <location>에러 페이지 URL</location>  
</error-page>
```

- 404 라면?

```
<error-page>  
    <error-code>404</error-code>  
    <location>/error404Page.jsp</location>  
</error-page>
```

# exception

## ▶ error404Page.jsp

```
<%@ page contentType="text/html; charset=euc-kr" %>
<%
    response.setStatus (HttpServletResponse.SC_OK) ;
%>
<html>
    <head>
        <title>404 에러 처리 페이지</title>
    </head>
    <body>
        <h3>JSP 페이지가 존재하지 않습니다.</h3>
    </body>
</html>
```

## ▶ 정상 처리 Page로 지정

```
<%    response.setStatus (HttpServletResponse.SC_OK) ; %>
```

# exception

## ▶ 예외 종류별 에러 처리

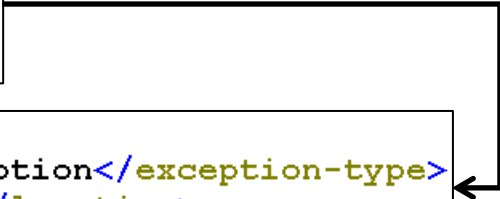
```
<%@ page contentType="text/html; charset=euc-kr" %>

<html>
  <head>
    <title>NullPointerException 발생 코드</title>
  </head>
  <body>
    <%
      String s = null;
      s.equals("NullPointerException");
    %>
  </body>
</html>
```

## ▶ web.xml

```
<error-page>
  <exception-type>예 외 종 류</exception-type>
  <location>에 러 페 이 지 URL</location>
</error-page>
```

```
<error-page>
  <exception-type>java.lang.NullPointerException</exception-type>
  <location>/errorNullPointerException.jsp</location>
</error-page>
```



A diagram consisting of a horizontal line with a downward-pointing arrow at its right end. This arrow points to the second `<error-page>` block, indicating that the general exception type is mapped to the specific `java.lang.NullPointerException`.

# exception

## ▶ errorNullPointerException.jsp

```
<%@ page contentType="text/html; charset=euc-kr" %>
<%
    response.setStatus (HttpServletResponse.SC_OK) ;
%>
<html>
    <head>
        <title>NullPointerException 에러 처리 페이지</title>
    </head>
    <body>
        <h3>NullPointerException이 발생하였습니다.</h3>
    </body>
</html>
```

## ▶ 에러 처리 page 용도

- 각 Server 제품군에서 제공하는 기본 에러 화면이 아닌
  - 각 Web Application에 Error Page 출력
- 사용자의 잘못된 요청이나 JSP Page 작성 중 발생할 수 있는
  - 오류에 대한 Page로서 활용
- 사용자가 당황하지 않도록... (친절한 Web Application의 시작)



# 오늘 숙제

▶ 집에서 해보자!