

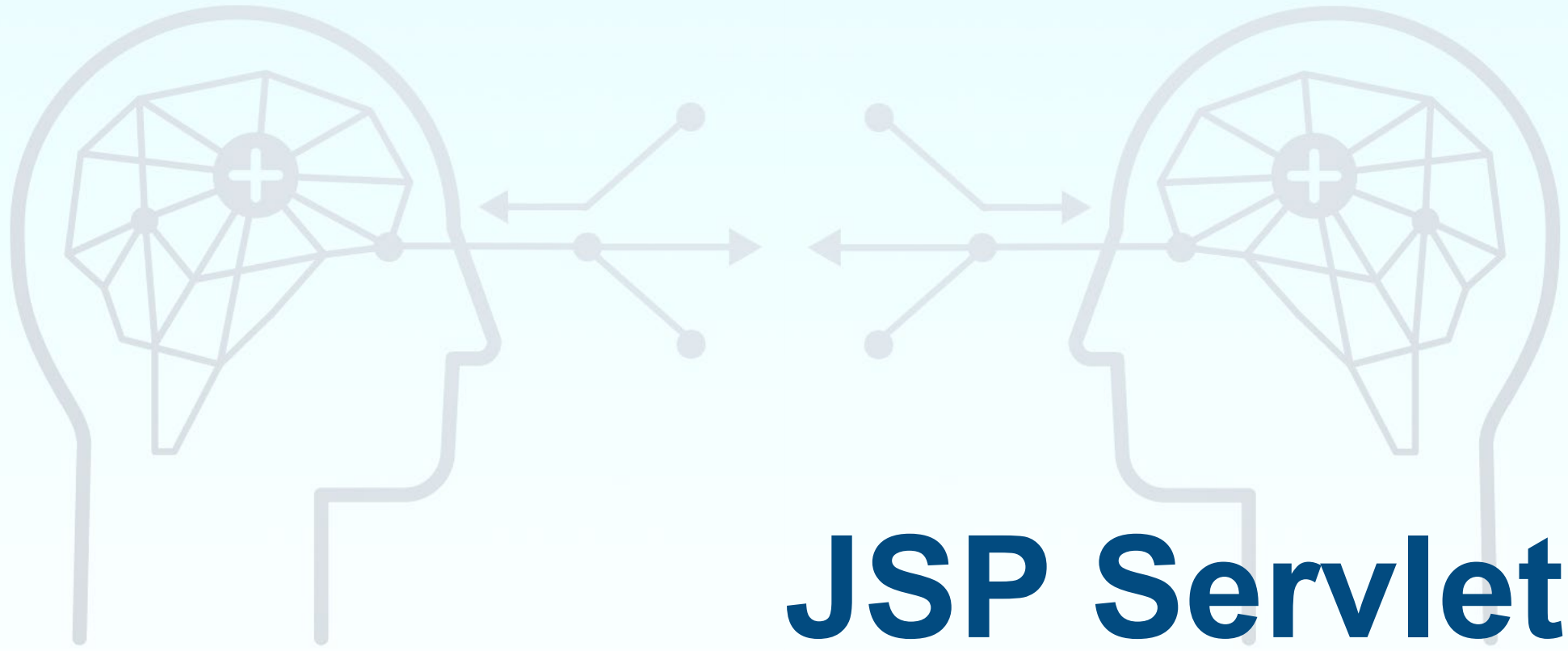


[한국폴리텍대학 성남캠퍼스 인공지능소프트웨어과]

JSP Servlet

2023. 5

김 형 오



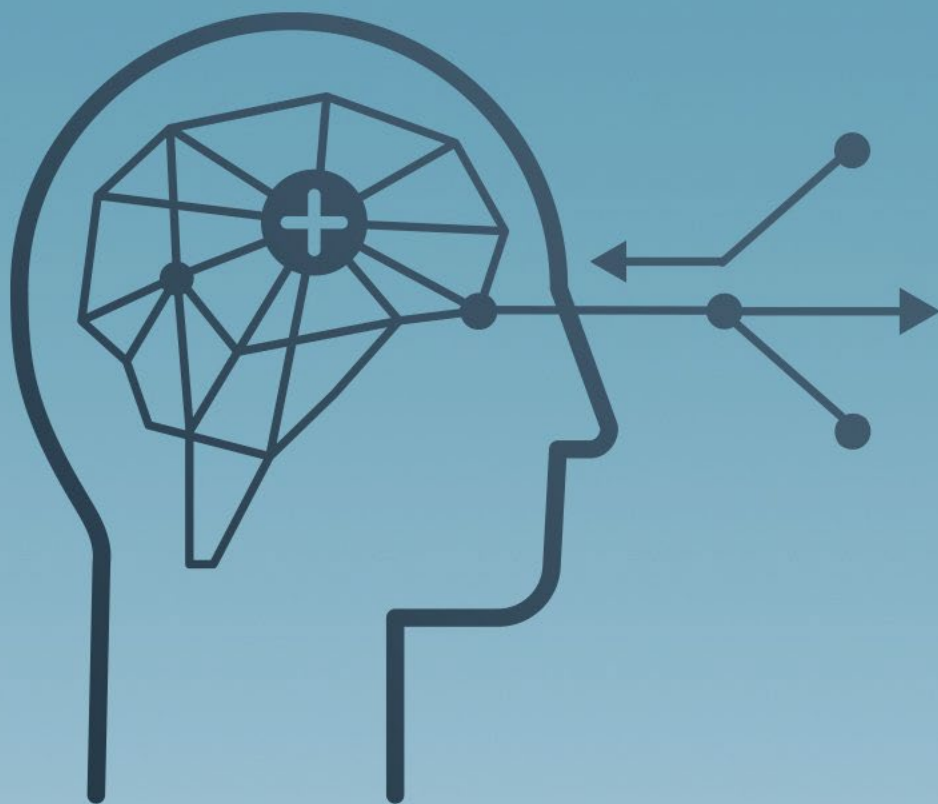
JSP Servlet

DEEP
LEARNING

MACHINE LEARNING BASED
ON ARTIFICIAL NEURAL NETWORKS

DEEP
LEARNING

MACHINE LEARNING BASED
ON ARTIFICIAL NEURAL NETWORKS



DEEP LEARNING

MACHINE LEARNING BASED
ON ARTIFICIAL NEURAL NETWORKS



목차

A table of Contents

#1, 문자 인코딩

#2, 쿠키와 세션

#3, Filter

#4, JSP

#5, 내장객체



Part 1,

문자 인코딩

참고문헌

<https://m.blog.naver.com/ycpiglet/222146759413>

<https://dololak.tistory.com/535>

<https://jeongdownon.medium.com/>

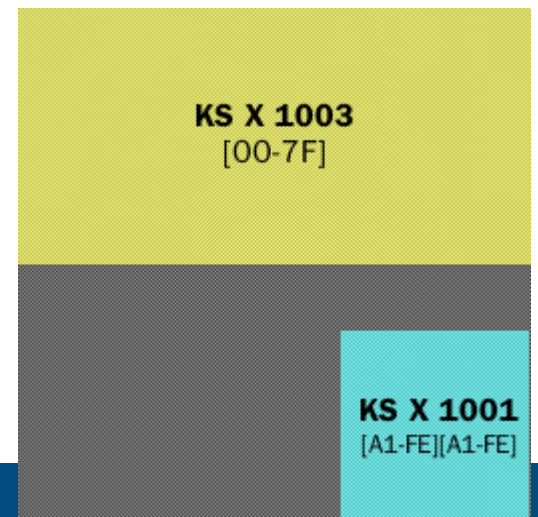
https://ko.wikipedia.org/wiki/%ED%99%95%EC%9E%A5_%EC%9C%A0%EB%8B%89%EC%8A%A4_%EC%BD%94%EB%93%9C

ASCII

- ANSI(American National Standards Institute)에서 표준화한 정보교환용 7bit 부호체계
- ASCII(American Standard Code for Information Interchange)

Decimal	Hexadecimal	Binary	Octal	Char	Decimal	Hexadecimal	Binary	Octal	Char	Decimal	Hexadecimal	Binary	Octal	Char
0	0	0	0	[NULL]	48	30	110000	60	0	96	60	1100000	140	'
1	1	1	1	[START OF HEADING]	49	31	110001	61	1	97	61	1100001	141	a
2	2	10	2	[START OF TEXT]	50	32	110010	62	2	98	62	1100010	142	b
3	3	11	3	[END OF TEXT]	51	33	110011	63	3	99	63	1100011	143	c
4	4	100	4	[END OF TRANSMISSION]	52	34	110100	64	4	100	64	1100100	144	d
5	5	101	5	[ENQUIRY]	53	35	110101	65	5	101	65	1100101	145	e
6	6	110	6	[ACKNOWLEDGE]	54	36	110110	66	6	102	66	1100110	146	f
7	7	111	7	[BELL]	55	37	110111	67	7	103	67	1100111	147	g
8	8	1000	10	[BACKSPACE]	56	38	111000	70	8	104	68	1101000	150	h
9	9	1001	11	[HORIZONTAL TAB]	57	39	111001	71	9	105	69	1101001	151	i
10	A	1010	12	[LINE FEED]	58	3A	111010	72	:	106	6A	1101010	152	j
11	B	1011	13	[VERTICAL TAB]	59	3B	111011	73	;	107	6B	1101011	153	k
12	C	1100	14	[FORM FEED]	60	3C	111100	74	<	108	6C	1101100	154	l
13	D	1101	15	[CARRIAGE RETURN]	61	3D	111101	75	=	109	6D	1101101	155	m
14	E	1110	16	[SHIFT OUT]	62	3E	111110	76	>	110	6E	1101110	156	n
15	F	1111	17	[SHIFT IN]	63	3F	111111	77	?	111	6F	1101111	157	o
16	10	10000	20	[DATA LINK ESCAPE]	64	40	1000000	100	@	112	70	1110000	160	p
17	11	10001	21	[DEVICE CONTROL 1]	65	41	1000001	101	A	113	71	1110001	161	q
18	12	10010	22	[DEVICE CONTROL 2]	66	42	1000010	102	B	114	72	1110010	162	r
19	13	10011	23	[DEVICE CONTROL 3]	67	43	1000011	103	C	115	73	1110011	163	s
20	14	10100	24	[DEVICE CONTROL 4]	68	44	1000100	104	D	116	74	1110100	164	t
21	15	10101	25	[NEGATIVE ACKNOWLEDGE]	69	45	1000101	105	E	117	75	1110101	165	u
22	16	10110	26	[SYNCHRONOUS IDLE]	70	46	1000110	106	F	118	76	1110110	166	v
23	17	10111	27	[ENG OF TRANS. BLOCK]	71	47	1000111	107	G	119	77	1110111	167	w
24	18	11000	30	[CANCEL]	72	48	1001000	110	H	120	78	1111000	170	x
25	19	11001	31	[END OF MEDIUM]	73	49	1001001	111	I	121	79	1111001	171	y
26	1A	11010	32	[SUBSTITUTE]	74	4A	1001010	112	J	122	7A	1111010	172	z
27	1B	11011	33	[ESCAPE]	75	4B	1001011	113	K	123	7B	1111011	173	{
28	1C	11100	34	[FILE SEPARATOR]	76	4C	1001100	114	L	124	7C	1111100	174	
29	1D	11101	35	[GROUP SEPARATOR]	77	4D	1001101	115	M	125	7D	1111101	175	}
30	1E	11110	36	[RECORD SEPARATOR]	78	4E	1001110	116	N	126	7E	1111110	176	~
31	1F	11111	37	[UNIT SEPARATOR]	79	4F	1001111	117	O	127	7F	1111111	177	[DEL]
32	20	100000	40	[SPACE]	80	50	1010000	120	P					
33	21	100001	41	!	81	51	1010001	121	Q					
34	22	100010	42	"	82	52	1010010	122	R					
35	23	100011	43	#	83	53	1010011	123	S					
36	24	100100	44	\$	84	54	1010100	124	T					
37	25	100101	45	%	85	55	1010101	125	U					
38	26	100110	46	&	86	56	1010110	126	V					
39	27	100111	47	'	87	57	1010111	127	W					
40	28	101000	50	(88	58	1011000	130	X					
41	29	101001	51)	89	59	1011001	131	Y					
42	2A	101010	52	*	90	5A	1011010	132	Z					
43	2B	101011	53	+	91	5B	1011011	133	[
44	2C	101100	54	,	92	5C	1011100	134	\					
45	2D	101101	55	-	93	5D	1011101	135]					
46	2E	101110	56	.	94	5E	1011110	136	^					
47	2F	101111	57	/	95	5F	1011111	137	_					

- 확장 유닉스 코드(Extended Unix Code, EUC)란 한국어, 중국어, 일본어 문자 전산화에 주로 사용되는 **8비트 문자 인코딩** 방식이다.
- EUC의 구조는 ISO 2022 표준에 기반하고 있다.
- 대한민국의 인터넷 환경에서 광범위하게 쓰이고 있어 친숙한 EUC-KR은 이 인코딩 방식을 사용하여 한글 등 한국어에서 사용되는 문자를 표현한 것이다. 사용빈도는 다르지만 중화인민공화국에서는 EUC-CN, 중화민국(대만)에서는 EUC-TW, 일본에서는 EUC-JP 등의 인코딩 방식이 존재한다.
- **EUC-KR은 KS X 1001와 KS X 1003을 사용하는 8비트 문자 인코딩**, EUC의 일종이며 대표적인 한글 완성형 인코딩이기 때문에 보통 완성형이라고 불린다.



UNICODE

- 전 세계의 모든 문자를 컴퓨터에서 일관되게 표현하고 다룰수 있도록 설계된 산업 표준
- 유니코드는 글자와 코드가 1:1 매핑되어 있는 '코드표' 이다

유니코드 영역 - 위키백과, 우리 모두의 백과사전 (wikipedia.org)

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040	 	Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	\$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	((72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051))	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[[123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174	|	
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135]]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177		DEL

UTF-8

- UTF-8은 UNICODE를 인코딩 하는 방식이다.
- 가변인코딩방식(글자마다 Byte길이가 다르다.)

	AC0	AC1	AC2	AC3	AC4	AC5	AC6	AC7	AC8	AC9	ACA	ACB	ACC	ACD	ACE	ACF
0	가	감	갠	갬	갈	각	갯	거	검	겐	갸	결	격	겜	고	곰
1	각	갑	갸	갯	갬	갯	갬	거	검	겐	갸	결	격	겜	곡	곱
2	각	갑	갸	갯	갬	갯	갬	거	검	겐	갸	결	격	겜	곡	곱
3	갸	갯	갬	갯	갬	갯	갬	거	검	겐	갸	결	격	겜	곡	곱
4	간	갸	갯	갬	갯	갬	갯	갬	건	갸	갯	갬	계	갸	곤	갸
5	갸	갯	갬	갯	갬	갯	갬	건	갸	갯	갬	계	갸	갯	곤	갸
6	갸	갯	갬	갯	갬	갯	갬	건	갸	갯	갬	계	갸	갯	곤	갸

'가'의 코드포인트 AC00

코드값의 자릿수	범위	첫 바이트	둘째 바이트	셋째 바이트	넷째 바이트	다섯째 바이트	여섯째 바이트
7비트	0-127	0xxxxxxx					
11비트	128-2,047	110xxxxx	10xxxxxx				
16비트	2,048-65,535	1110xxxx	10xxxxxx	10xxxxxx			
21비트	65,536-2,097,151	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx		
26비트		111110xx	10xxxxxx	10xxxxxx	10xxxxxx	10xxxxxx	
31비트		1111110x	10xxxxxx	10xxxxxx	10xxxxxx	10xxxxxx	10xxxxxx

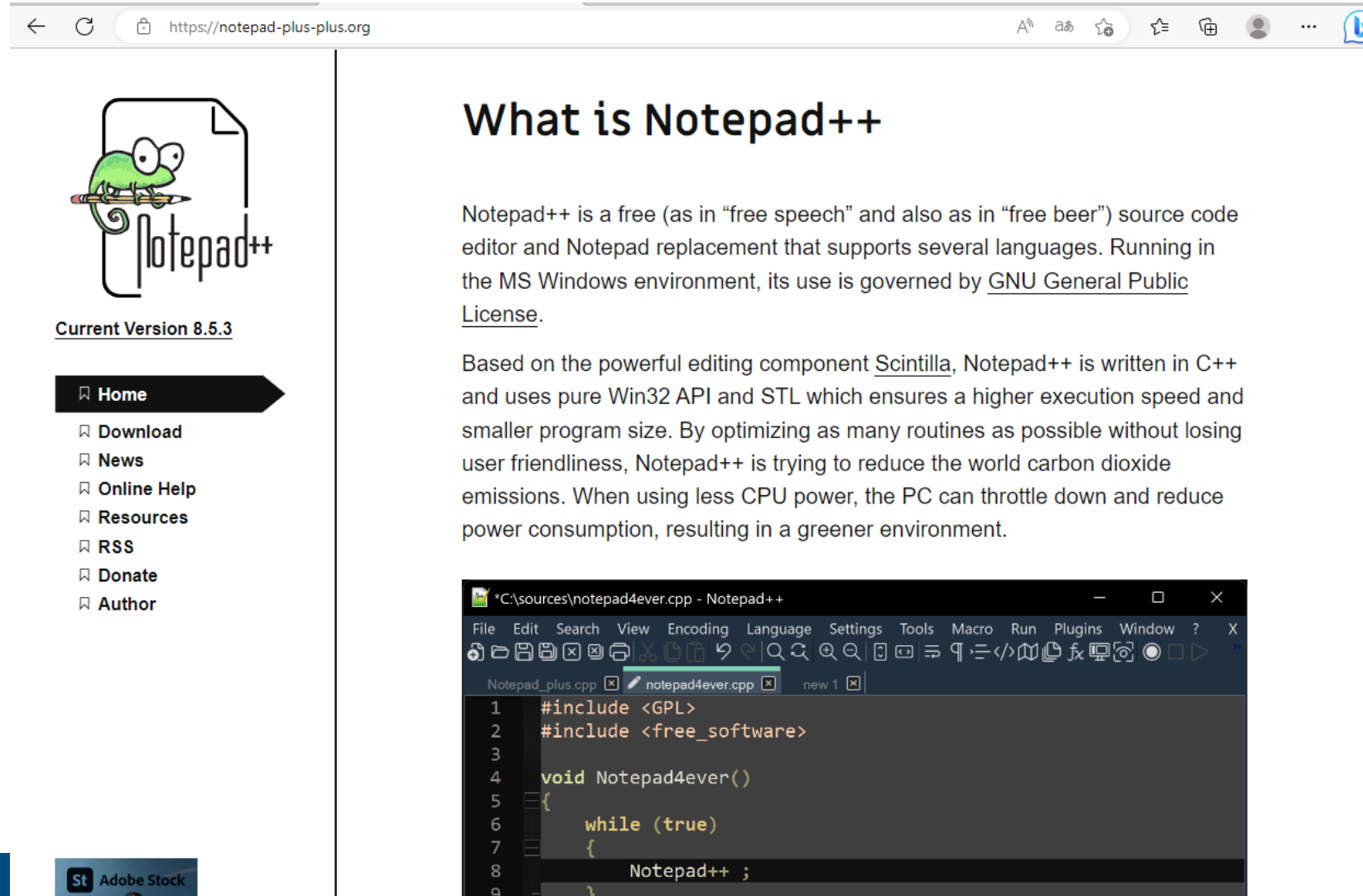
즉 유니코드hex값이 AC00 인 한글 "가"의 binary 값
은 1010110000000000 이것을 3등분 해서 utf-8 인코딩 알고리즘을 적용하면

11101010
10110000
10000000

//첫 바이트는 1110 시작 (hex값: EA)
//나머지 바이트는 10 시작 (hex값: BO)
//나머지 바이트는 10 시작 (hex값: 80)

Notepad++

<https://notepad-plus-plus.org/>



The screenshot shows the Notepad++ website at <https://notepad-plus-plus.org/>. The website features a logo of a green frog on a notepad, the text "Notepad++", and "Current Version 8.5.3". A navigation menu on the left includes links to Home, Download, News, Online Help, Resources, RSS, Donate, and Author. The main content area is titled "What is Notepad++" and contains two paragraphs of text. The first paragraph states that Notepad++ is a free source code editor and Notepad replacement that supports several languages, governed by the GNU General Public License. The second paragraph explains that it is based on the Scintilla editing component, written in C++, and uses pure Win32 API and STL for higher execution speed and smaller program size, aiming to reduce carbon dioxide emissions by using less CPU power.

Below the text, there is a screenshot of the Notepad++ application window. The window title is "C:\sources\notepad4ever.cpp - Notepad++". The menu bar includes File, Edit, Search, View, Encoding, Language, Settings, Tools, Macro, Run, Plugins, Window, and Help. The toolbar contains various icons for file operations and editing. The editor shows a C++ file named "notepad4ever.cpp" with the following code:

```
1 #include <GPL>
2 #include <free_software>
3
4 void Notepad4ever()
5 {
6     while (true)
7     {
8         Notepad++ ;
9     }
```



Part 2,

쿠키와 세션

참고문헌

<https://dololak.tistory.com/535>

쿠키의 배경

- HTTP는 stateless protocol(무상태 프로토콜)이다.
- 쿠키는 HTTP 에서 상태를 유지하기 위한 기술이다.

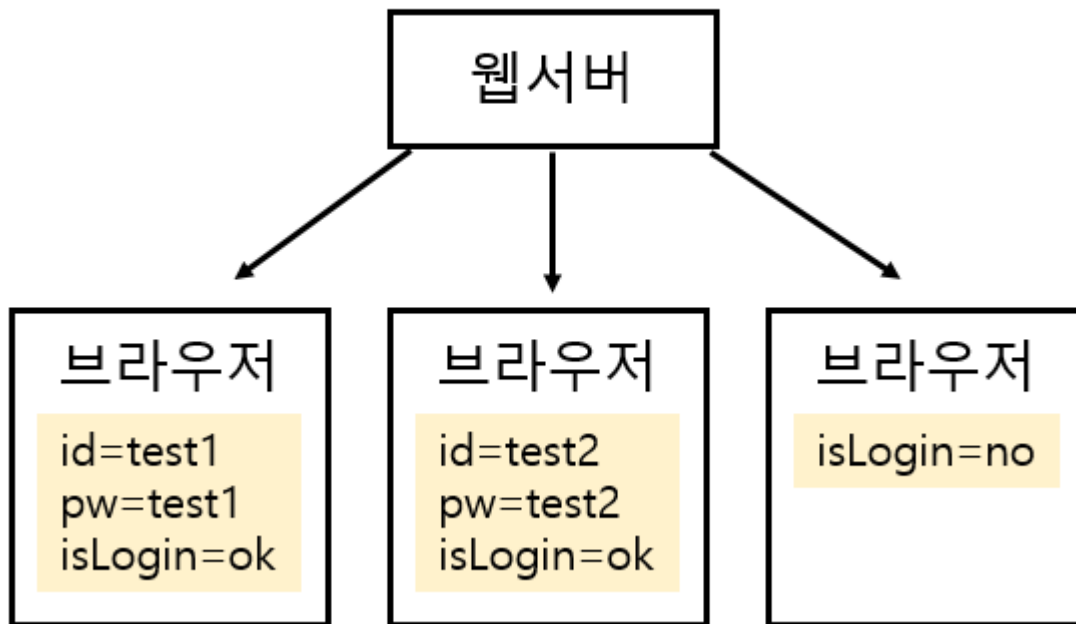


- 최초 페이지에서 로그인하고 여러 페이지에 걸쳐 로그인 인증정보가 유지되어야 하는 경우(로그인 페이지에서 로그인 하였는지 검증시)
- 쇼핑몰 장바구니에 담긴 아이템 정보들이 다른 페이지로 이동시에도 유지되어야 하는 경우(새로고침이나 다른 항목으로 페이지 이동시 장바구니 데이터가 날아가지 않기 위함)
- 회원가입시 여러 단계에 걸쳐 가입이 이루어지는 경우 이전 단계 페이지의 정보들이 유지되어야 하는 경우(이전 단계를 거쳤는지 검증시)

쿠키 원리

- 쿠키 - 클라이언트의 상태 데이터를 서버가 아닌 클라이언트측 브라우저에 저장
- 브라우저에 저장된 쿠키 데이터는 개인정보 유출에 취약

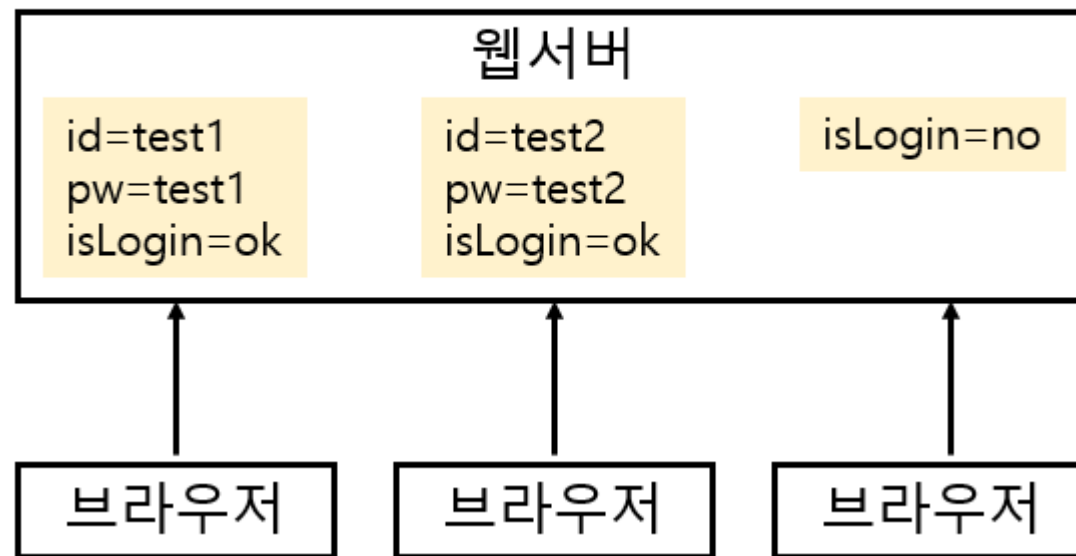
쿠키 - 데이터를 클라이언트 측에 저장



세션 원리

- 세션은 쿠키와는 반대로 데이터를 서버에 저장해 두고 브라우저별로 구분하여 각각 생성
- 데이터 크기에 제한이 있었던 쿠키와는 다르게 서버 성능이 되는한 마음대로 데이터를 저장할 수 있다.

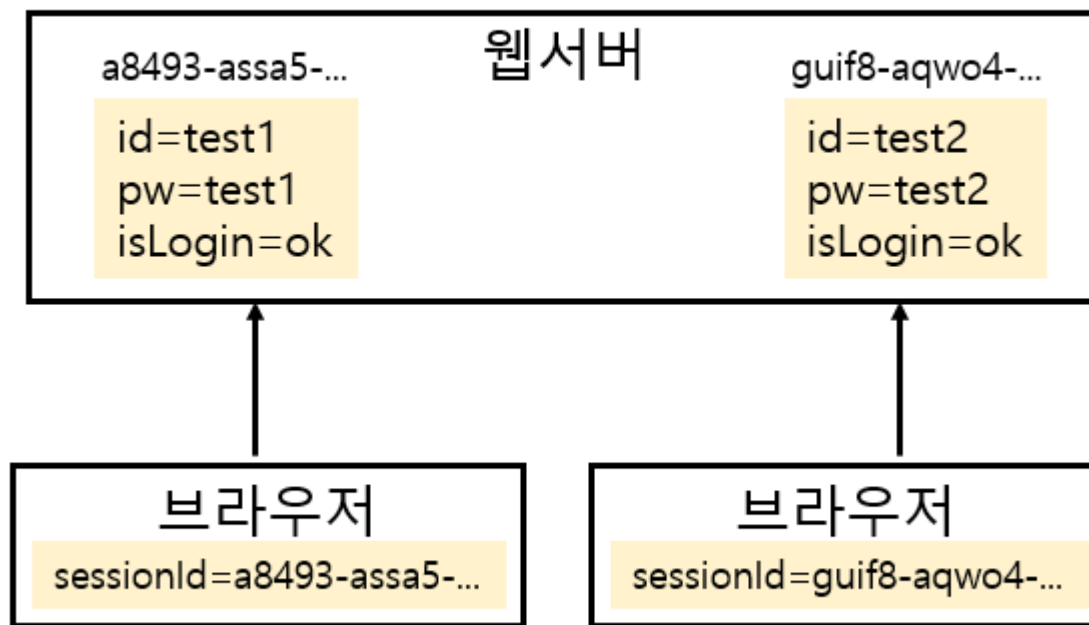
세션 – 데이터를 서버 측에 저장



쿠키-세션 활용

- 서버에 저장되어있는 자신의 세션을 구분하기 위한 세션키값을 쿠키로 저장하여 사용

세션 - 데이터를 서버 측에 저장



쿠키 - 세션을 구분하기 위한 키를 저장

Part 2 쿠키 예제 - 1

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    int cnt = 0;
    Cookie[] cookies = request.getCookies();
    for(int i = 0; cookies != null && i < cookies.length; i++) {
        if(cookies[i].getName().equals("count")) {
            cnt = Integer.parseInt(cookies[i].getValue());
            break;
        }
    }

    cnt++;
    Cookie c = new Cookie("count", cnt + "");
    response.addCookie(c);

    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    out.print("<h1>방문 횟수: " + cnt);
    out.close();
}
```

Part 2 쿠키 예제 - 2

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    request.setCharacterEncoding("euc-kr");
    response.setContentType("text/html;charset=EUC-KR");
    PrintWriter out = response.getWriter();

    Cookie[] cookies = request.getCookies();
    if(cookies != null) {
        for(Cookie cookie : cookies) {
            out.println("cookie :"+cookie.getName()+":"+cookie.getValue()+"<br/>");
        }
    }
    out.println("<form method='post' action='CookieTestServlet'>");
    out.println("name<input type='text' name='name'/>");
    out.println("value<input type='text' name='value'/>");
    out.println("<input type='submit'/>");
    out.println("</form>");
}

/**
 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
 */
protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    String name = request.getParameter("name");
    String value = request.getParameter("value");

    Cookie cookie = new Cookie(name, value);
    response.addCookie(cookie);
    response.sendRedirect("CookieTestServlet");
}
```



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

```
    HttpSession session = request.getSession();
```

```
        Integer count = (Integer)session.getAttribute("count");
```

```
        if (session.getAttribute("count") == null) {
```

```
            session.setAttribute("count", 1);
```

```
        } else {
```

```
            session.setAttribute("count", count + 1);
```

```
        }
```

```
        PrintWriter out = response.getWriter();
```

```
        out.println("count : " + session.getAttribute("count"));
```

```
    }
```



Part 3,

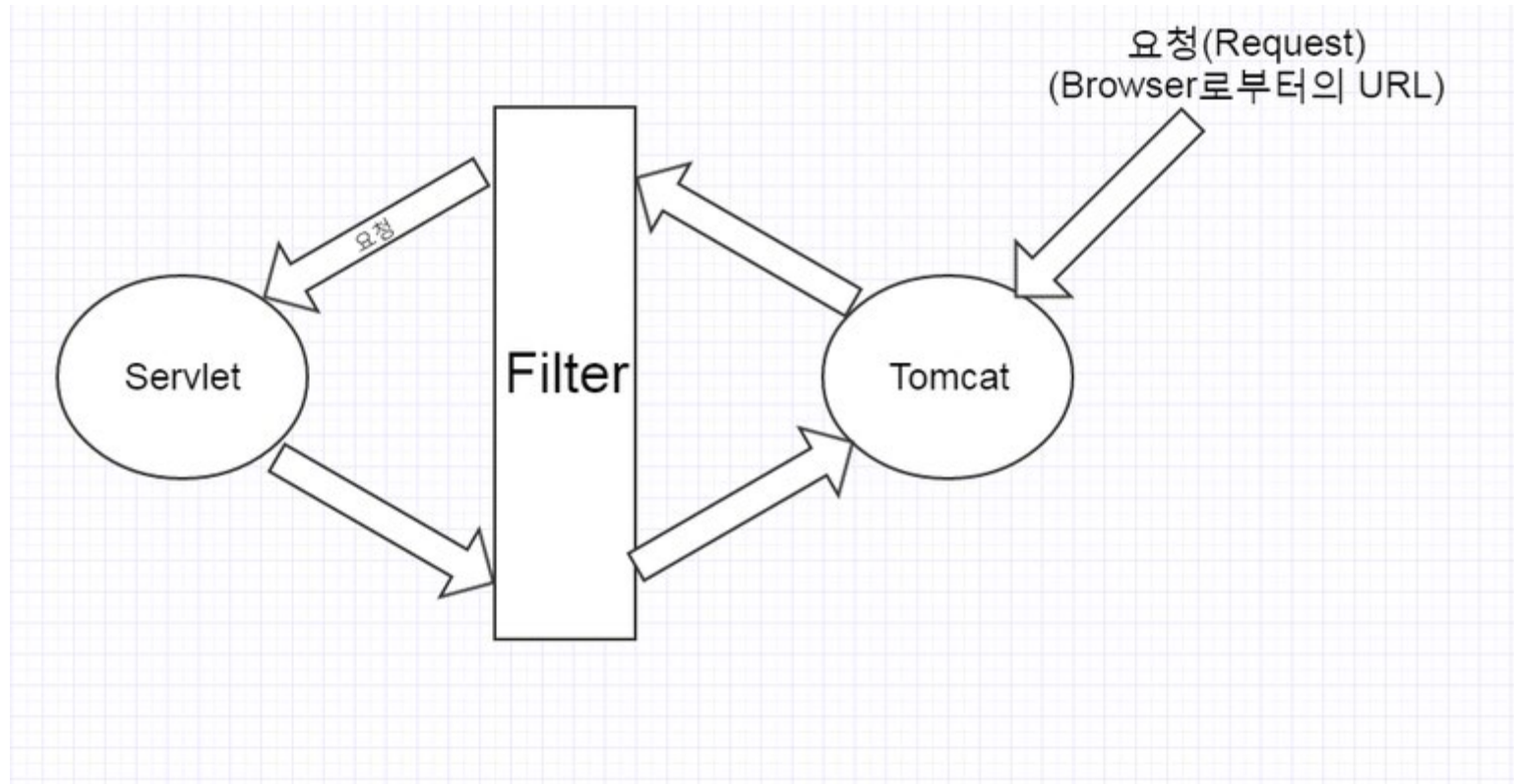
Filter

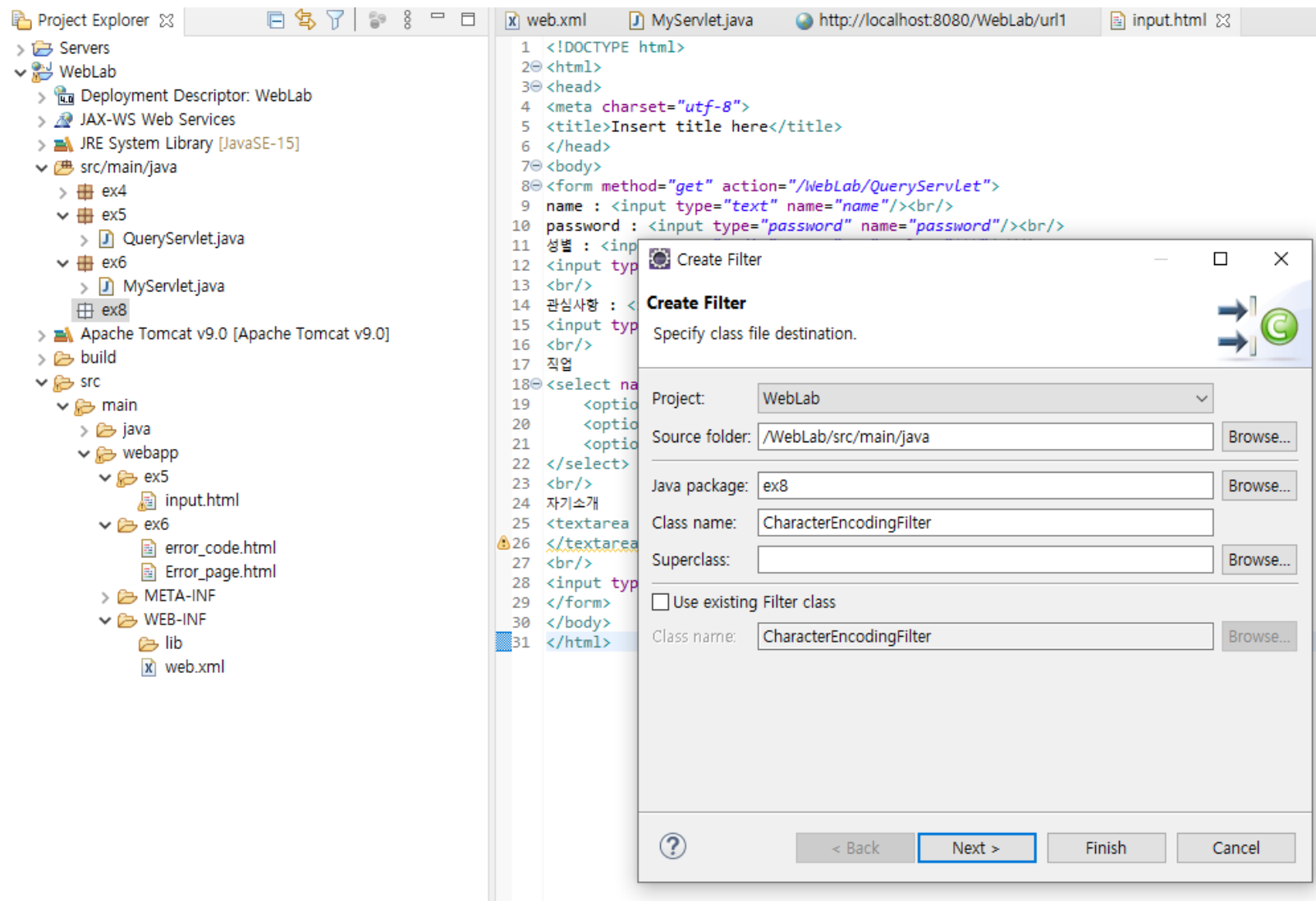
참고문헌

<https://minwoohi.tistory.com/60>

Filter

- 톰캣(Tomcat)에서 url 요청을 받으면 web.xml에 등록되어 있는 url pattern에 대해서는 해당 서블릿이 수행돼 그 결과를 response 객체에 담아 톰캣에 반환.
- 이때 필터(Filter)를 두어 **서블릿들에서 중복되는 로직을 한번에 처리**할 수 있다. 필터에는 주로 인코딩, 로그인 체커 로직을 담는다.





Filter

Create Filter

Enter servlet filter deployment descriptor specific information

Name: CharacterE
Description:
Initialization parameters:

Name

Filter mappings:

URL Pattern / Servlet name	Dispatchers
/CharacterEncodingFilter	

☐ Asynchronous Support

OK Cancel Finish

Edit Filter Mapping

☐ Servlet ☒ URL pattern

Pattern: /CharacterEncodingFilter

Select dispatchers

☐ REQUEST ☐ FORWARD
☐ INCLUDE ☐ ERROR

OK Cancel

Create Filter

Enter servlet filter deployment descriptor specific information

Name: CharacterE
Description:
Initialization parameters:

Name

Filter mappings:

URL Pattern / Servlet name	Dispatchers
/CharacterEncodingFilter	

☐ Asynchronous Support

OK Cancel Finish

Edit Filter Mapping

☐ Servlet ☒ URL pattern

Pattern: /*

Select dispatchers

☐ REQUEST ☐ FORWARD
☐ INCLUDE ☐ ERROR

OK Cancel

Filter

CharacterEncodingFilter.java

```
public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
throws IOException, ServletException {
    System.out.println("Filter begin....");
    request.setCharacterEncoding("UTF-8");
    response.setCharacterEncoding("UTF-8");

    chain.doFilter(request, response);
    System.out.println("Filter end....");
}
```

FilterTestServlet.java

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
    System.out.println("Servlet begin....");
    System.out.println("Servlet end....");
}
```



Part 4,

JSP

참고문헌

https://dinfree.com/lecture/backend/javaweb_2.2.html

JSP는 서블릿의 화면처리 어려움을 해결하기 위해 나왔으며 HTML 과 데이터를 조합하기 위해 다음과 같은 주요 기능적 특징을 가진다.

- HTML 페이지에 자바코드를 직접 사용
- 서블릿 컨테이너에 의해 관리되는 내장객체들의 라이프사이클을 이용한 페이지간 속성관리
- 커스텀 태그 기술을 사용한 코드의 태그화(action, JSTL등)
- EL(Expression Language)를 통한 데이터 표현

지시어(Standard directives)

지시어(Directives)는 해당하는 JSP 파일의 속성을 기술하는 곳으로, JSP 컨테이너에게 해당 페이지를 어떻게 처리해야 하는지 전달하기 위한 내용을 담고 있다. 지시어는 크게 page, include, taglib으로 나눌 수 있으며, 각각의 속성이 다르다.

지시어의 기본 형식은 다음과 같다.

```
<%@ directive attribute="value" %>
```

page 지시어

page 지시어는 현재의 JSP 페이지를 컨테이너에서 처리(서블릿으로 변환)하는 데 필요한 각종 속성을 기술하는 부분으로, 소스코드 맨 앞에 위치하며 이클립스에서 jsp 파일을 생성할때 자동으로 포함되며 구문과 일반적인 사용에는 다음과 같다.

```
<%@ page page_directive_attr_list %>
```

```
<%@ page language="java" contentType="text/html; charset=UTF-8"  
    pageEncoding="UTF-8" import="java.util.*" errorPage="error.jsp"%>
```

- **language** : 현재 페이지의 스크립트 언어를 지정하는 속성, 스펙상 다른 언어도 가능하지만 당연히 java 를 기본으로 함.
- **contentType** : 현재 페이지의 파일 형식을 지정하는 속성, 클라이언트 요청에 응답할때 전달하는 HTTP 헤더 정보가 된다.
- **pageEncoding** : jsp 파일을 컨테이너가 처리할때 사용하는 캐릭터 인코딩을 지정하는 속성, 올바른 한글 처리를 위해서는 UTF-8 로 지정.
- **import** : jsp 파일내에서 자바코드(스크립트릿)를 직접 사용하는 경우 일반 자바 코드와 마찬가지로 클래스에 대한 패키지 import 가 필요하다.
- **errorPage** : 현재 jsp 요청 처리중에 에러가 발생하는 경우 서버 에러를 클라이언트에 전달하지 않고 별도의 페이지에서 처리하기 위한 속성. 매 페이지에 에러 페이지 설정을 넣는것 보다는 서버 설정을 사용하는 것이 권장됨.

include 지시어

다른 파일을 포함하기 위한 지시어로 include 지시어가 사용된 위치에 해당 파일(html, jsp)을 불러온다. 컨테이너에서는 포함된 파일들을 하나로 처리하며 java 소스를 생성한 뒤 서블릿으로 컴파일하게 된다.

따라서 포함되는 파일의 경우 단독 실행(해당 파일을 직접 요청해서 사용)을 하지 않는다고 하면 개별 구성요소(page 지시어, html 기본 태그 구성요소들)들을 갖출 필요가 없다.

include 액션은 include 지시어와 유사하게 동작하지만 지시어와 달리 각각의 파일들을 실행 시점에서 요청해 결과만 포함하는 구조이다.

include 지시어는 원하는 위치에 자유롭게 사용할 수 있으며 사용예는 다음과 같다.

```
<%@ include file="relativeURLspec" %>
```

- file: 포함하고자 하는 파일의 상대 URL 경로

```
<%@ include file="header.jsp" %>  
<%@ include file="body.jsp" %>  
<%@ include file="footer.html" %>
```

taglib 지시어

taglib 지시어는 jsp 의 태그 확장 메커니즘인 커스텀 태그를 사용하기 위한 지시어 이다. 커스텀 태그과 관련해서는 뒤에서 자세히 살펴보도록 한다.

taglib 지시어의 구문과 사용예는 다음과 같다.

```
<%@ taglib ( uri="tagLibraryURI" | tagdir="tagDir" ) prefix="tagPrefix" %>
```

- uri: 태그라이브러리 URI로 태그를 정의하고 있는 .tld 파일의 위치
- tagdir: 태그파일로 태그 구현시 태그 파일들이 있는 위치
- prefix: 해당 태그를 구분해서 사용하기 위한 접두어

```
<%@ taglib tagdir="/WEB-INF/tags" prefix="m" %>
...
<h2><m:printData /></h2>
```

JSP 예제

head.jsp

```
<%@ page language="java" contentType="text/html; charset=EUC-KR"  
    pageEncoding="EUC-KR"%>  
<CENTER>  
<FONT COLOR="GREEN">  
<H3>include 지시문 예제... </H3>  
</FONT>  
</CENTER>
```


JSP 예제

main.jsp

```
<%@ page language="java" contentType="text/html; charset=EUC-KR"
    pageEncoding="EUC-KR"%>
<%@ page import = "java.util.Date" %>
<%@ page session="false" %>

<%!
Date date;
String name;
String email;
%>

<%!
public int getLength(){
int len=email.length();
return len;
}
%>
<!DOCTYPE html>
<html>
<head>
<meta charset="EUC-KR">
<title>Insert title here</title>
</head>
```

JSP 예제

main.jsp

```
<body>
<h1>스크립트 태그 테스트</h1>
<%
date = new Date();
%>
현재 날짜 : <%=date.toLocaleString() %>
<br/>
<%@ include file="head.jsp" %>
<br/>
<%
name = request.getParameter("name");
email = request.getParameter("email");
%>
name : <%=name %><br/>
email : <%=email %><br/>
<%=date.getDate() %>일 입니다. 이 배열의 길이는<%=getLength() %>입니다.
</body>
</html>
```



Part 5,

내장객체

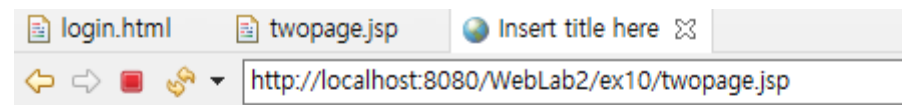
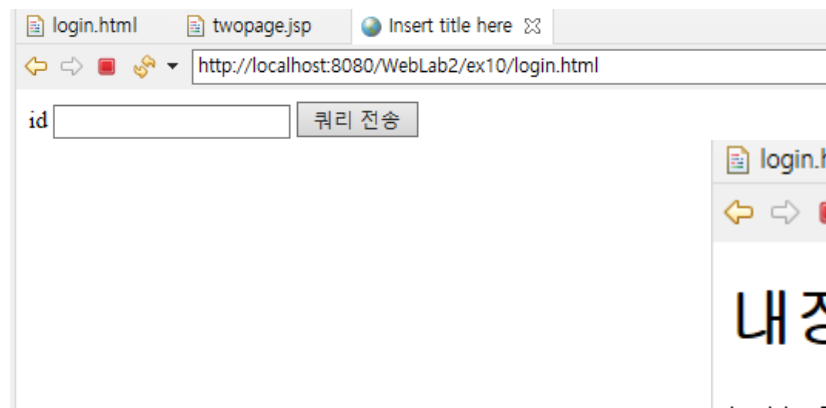
참고문헌

https://dinfree.com/lecture/backend/javaweb_2.2.html

내장객체 종류

내장객체는 기본적으로 다음 표의 9개를 말한다. 이들은 웹 애플리케이션 운영을 위해 다양한 목적에서 만들어진 것으로 JSP만을 위해 개발된것이 아니라 대부분 원래 서블릿에서 사용되던 클래스들이다(`javax.servlet.*` 패키지 사용). 각각의 내장객체에 대한 자세한 사항은 API 문서를 참조하기 바란다.

내장객체	타입	주요 제공 기능
out	<code>javax.servlet.jsp.JspWriter</code>	클라이언트(웹브라우저)로 출력 전달
request	<code>javax.servlet.http.HttpServletRequest</code>	HTTP 요청 처리
response	<code>javax.servlet.http.HttpServletResponse</code>	HTTP 응답 처리
config	<code>javax.servlet.ServletConfig</code>	서블릿 관련 정보 처리
application	<code>javax.servlet.ServletContext</code>	웹애플리케이션 정보 처리
session	<code>javax.servlet.http.HttpSession</code>	세션 정보 처리
pageContext	<code>javax.servlet.jsp.PageContext</code>	다른 내장 객체의 참조나 forward에 사용
page	<code>java.lang.Object</code>	현재 JSP 문서 정보 처리, 현재 페이지의 서블릿 객체를 가리킴
exception	<code>java.lang.Throwable</code>	예외 처리



두번째 페이지

request : null
session : hokim
application : hokim

login.html

```
<body>  
<form method="post" action="./onepage.jsp">  
id <input type="text" name="id">  
<input type="submit"/>  
</form>  
</body>
```

내장객체 예제

Onepage.jsp

```
<body>
<h1>내장 객체 테스트</h1>
<%
String id = request.getParameter("id");
request.setAttribute("id", id);
session.setAttribute("id", id);
application.setAttribute("id", id);
%>
<%= id%>로 로그인 하였습니다.<br/>
<a href="./twopage.jsp">이동</a>
</body>
```

내장객체 예제

twopage.jsp

```
<body>
<h1>두번째 페이지</h1>
<%
String requestid = (String)request.getAttribute("id");
String sessionid = (String)session.getAttribute("id");
String applicationid = (String)application.getAttribute("id");
%>
request : <%= requestid %><br/>
session : <%= sessionid %><br/>
application : <%= applicationid %><br/>
</body>
```

경청해주셔서 감사합니다.