

김 혜 경

topickim@naver.com

Web Component[Servlet&JSP]

학습내용

Server & Client 개요

Web 개발 기술 및 Web 실행 구조

Java Web Application

Web 개발 환경 구축

Servlet & JSP

Server & Client 개요

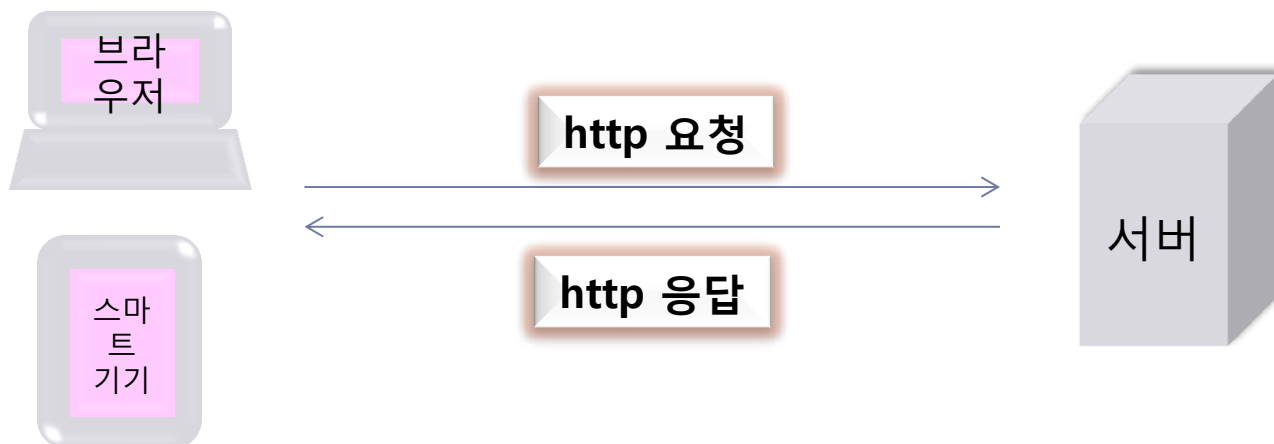
Server & Client 개요

1. Server & Client

1. Server - 네트워크상에서 서비스를 제공하는 System
2. Client - 네트워크상에서 서비스를 제공받는 System

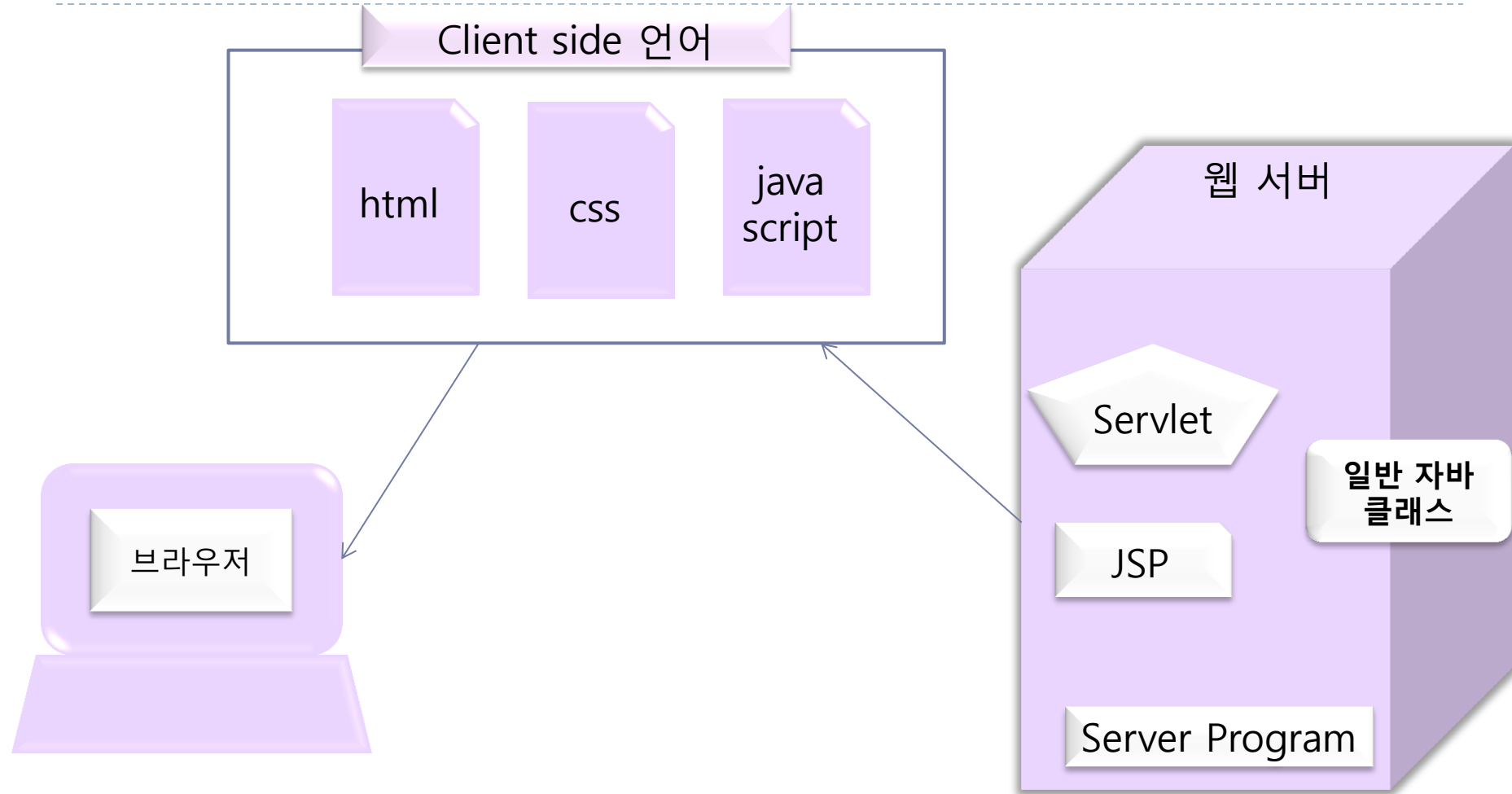
2. HTTP(HyperText Transfer Protocol)

1. protocol - 네트워크에 연결된 컴퓨터가 서로 통신(대화)하기 위한 규약
2. Web Server와 Client 통신 프로토콜



Web 개발 기술 및 Web 실행 구조

Web 개발 기술들



Web 개발 기술들

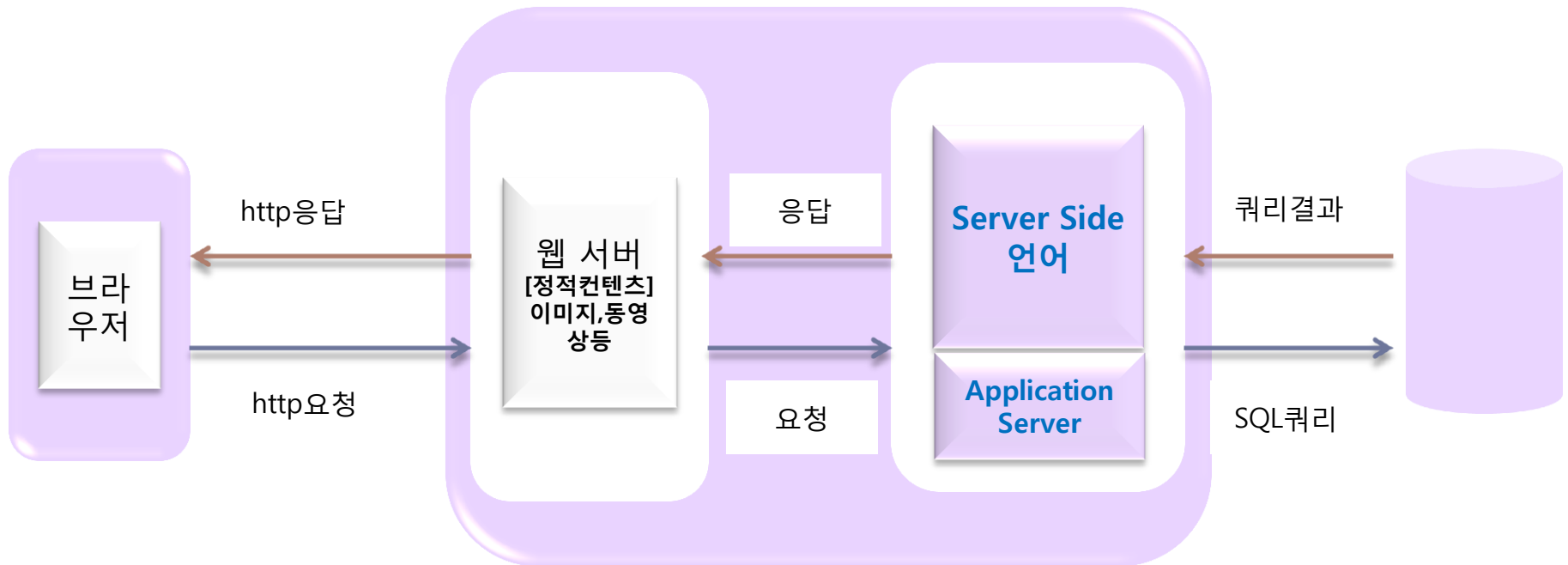
Client Side 언어

1. Client 브라우저에서 실행시키는 언어
2. 종류
 1. HTML : 페이지 내용 기술
 2. CSS : 디자인 정보 기술
 3. Java Script : 동적인 HTML 문서가 될수 있도록 지원하는 명령어

Server Side 언어

1. 서버에서 실행 후 Client에는 결과만 응답하게 되는 언어
2. JSP[Java], ASP, PHP등

Web 실행 구조



* Web Server 실행 구조

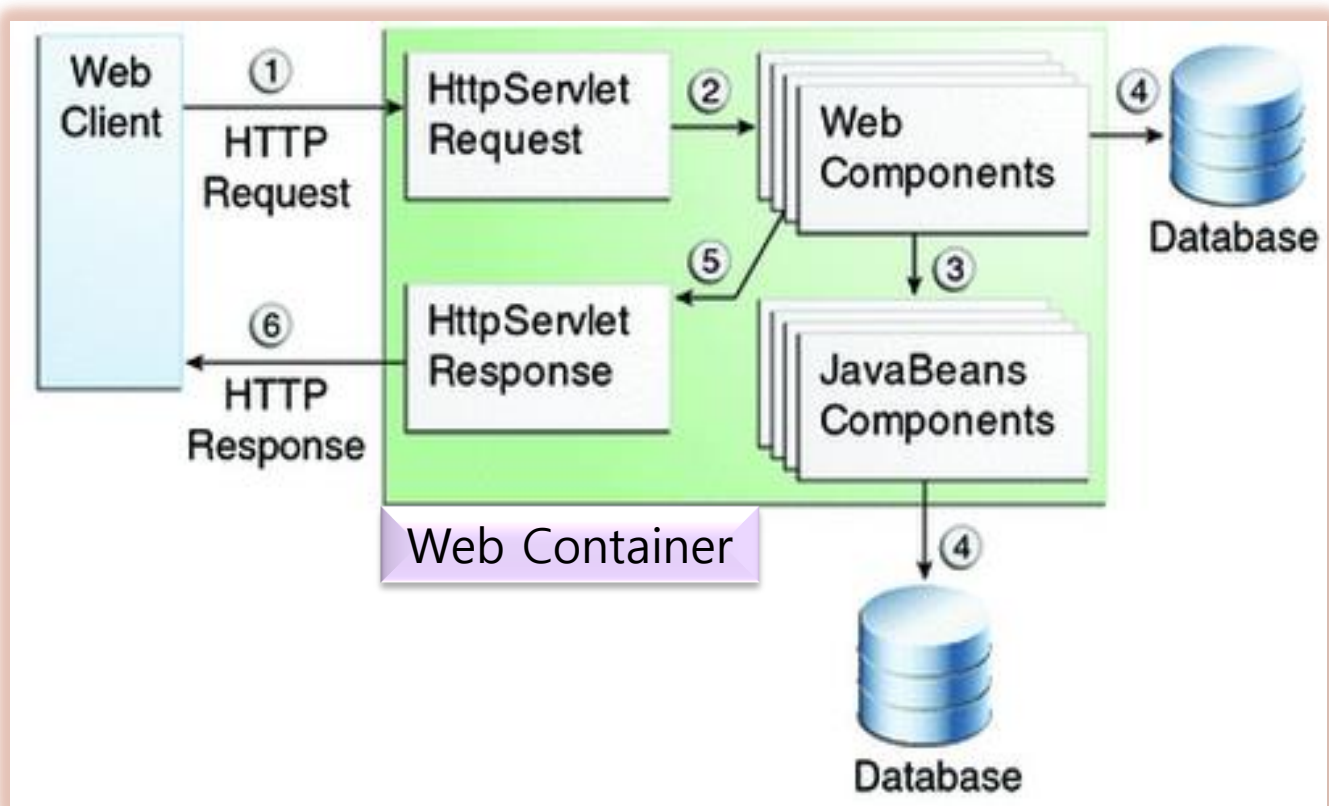
- 정적 content : 웹 서버로 부터 응답이 됨
- 동적 content : Web Application Server로 부터 응답이 됨

Java Web Application

Java Web Application[Servlet & JSP]

	Servlet	JSP[Java Server Pages]
공통점	<ol style="list-style-type: none">1. 웹 콘텐츠를 생성하기 위한 JavaEE 지원 스펙2. Web Container 내에서 실행3. client 요청을 thread로 처리	
차이점	<ol style="list-style-type: none">1. 확장자 - *.java2. 구성 : 자바 문법 + 경우에 따라 HTML tag3. Controller 로직 권장	<ol style="list-style-type: none">1. 확장자 - *.jsp2. 구성 : 자바 + HTML tag + JSP Scripting tag + JSP Action tag + EL + JSTL + CSS + java script3. 실행시 Web Container가 Servlet으로 자동 변환4. View 로직 권장

Java Web Application Request Handling



<http://docs.oracle.com/javaee/6/tutorial>

Web Application Server란?

▶ Web Application Server

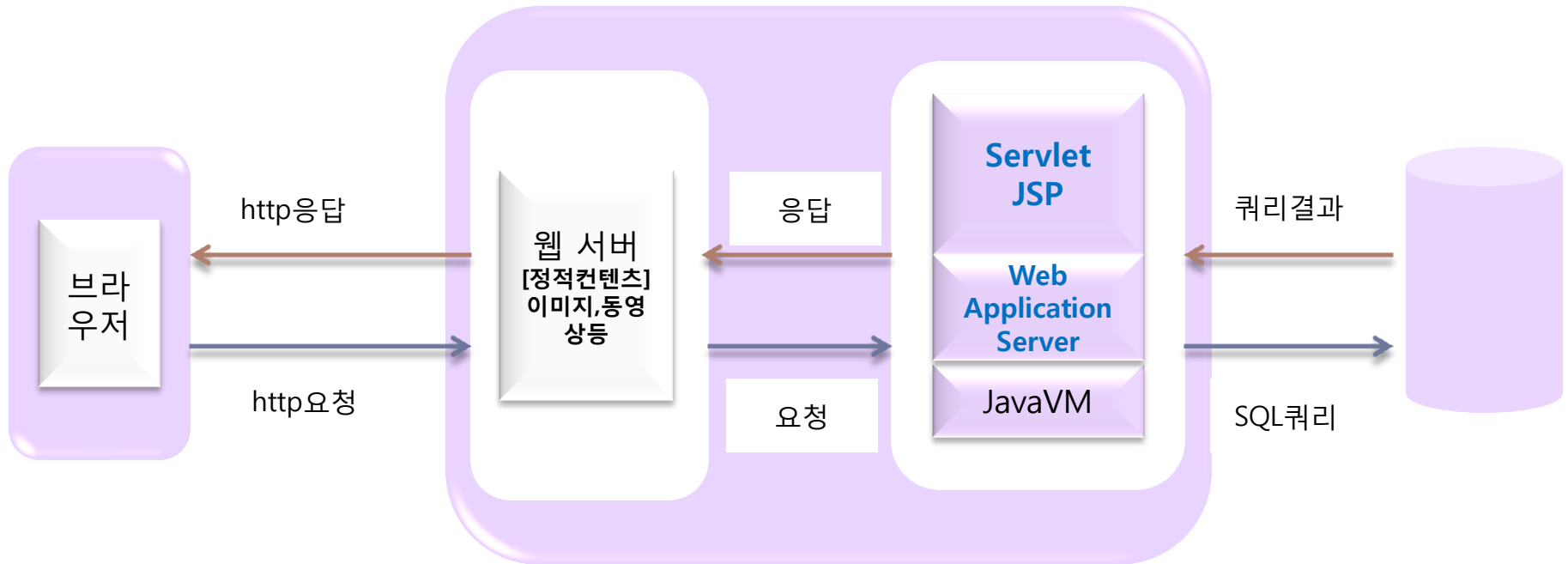
1. Web Container
2. JVM을 이용하여 Servlet & JSP 의 실행 환경 제공 및 life cycle 관리 및 서비스

1. 아파치 소프트웨어 재단

1. Apache Software Foundation
2. **오픈 소스 소프트웨어를 지원하는 비영리 단체**
3. 사용 Server
 1. Tomcat 8
 2. 아파치 소프트웨어재단에서 개발한 오픈소스 Application server



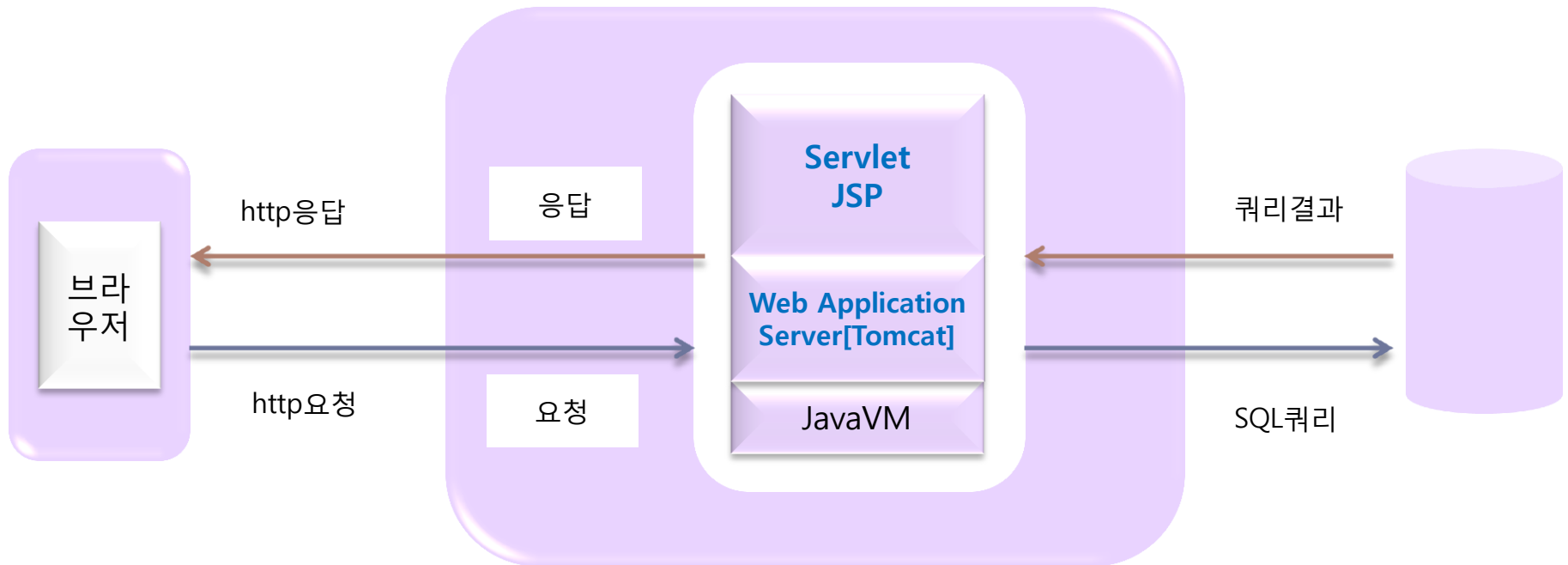
Web 실행 구조



* Servlet & JSP 실행 구조

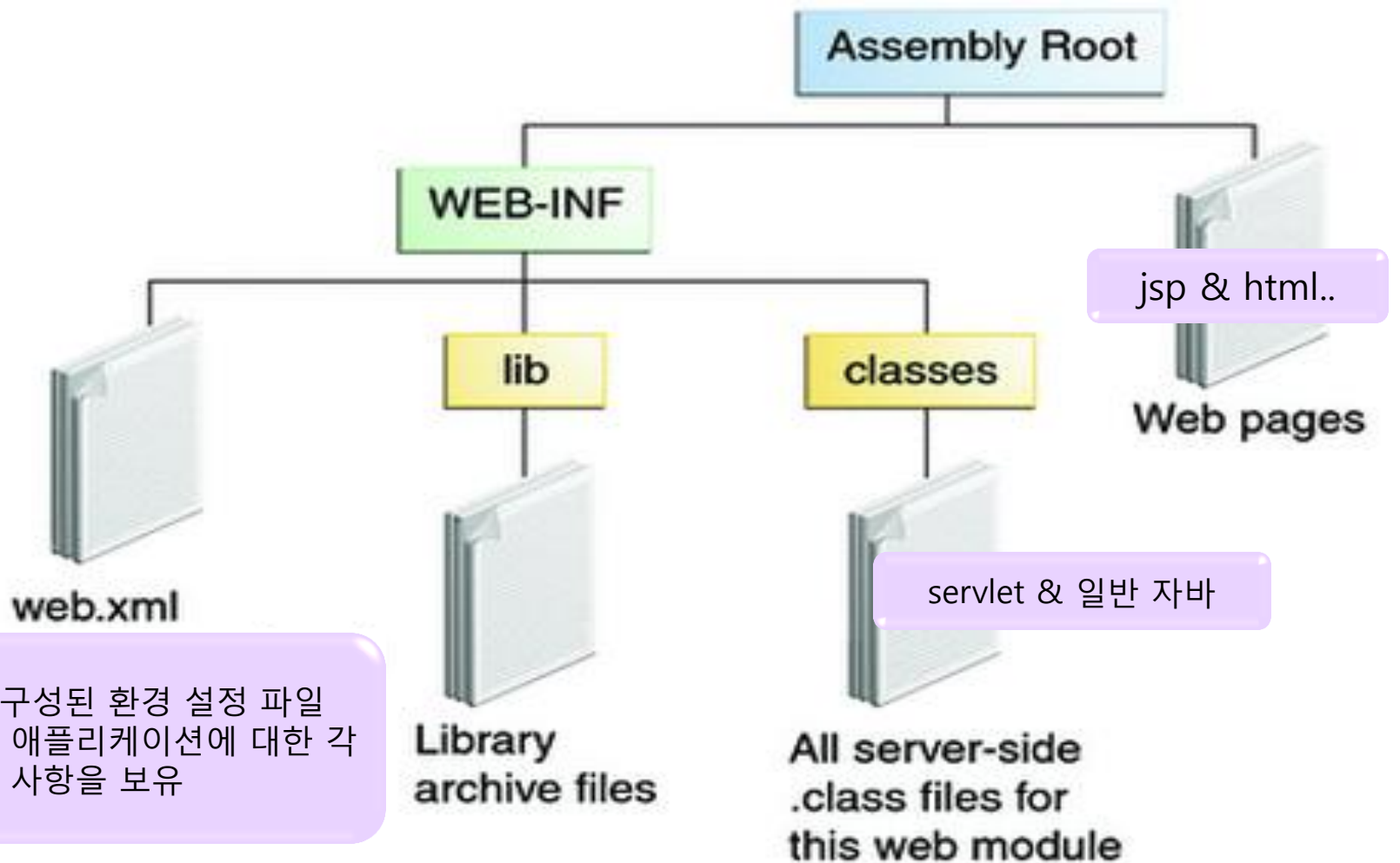
Java VM 상에서는 Application Server[Web Container]가 작동
Application Server는 Servlet & JSP 실행

Servlet & JSP 실행 구조



* Servlet & JSP 실행 구조
규모가 크지 않은 웹 시스템 구조

Web Module Structure : (구)sun의 실행 서버 스펙



Web 개발 환경 구축

HTTP 특징 - 인터넷과 www

1. 인터넷은 TCP/IP 기반의 네트워크가 전세계적으로 확대되어 하나로 연결된 'network of network'
2. www : 인터넷 기반의 서비스 중 하나

이름	프로토콜	포트	기능
www	http	80	웹 서비스
Email	SMTP/POP3/IMAP	25/110/114	이메일 서비스
FTP	ftp	21	파일 전송 서비스
telnet	telnet	23	원격 로그인
DNS	DNS	83	도메인 이름 변환 서비스
News	NNTP	119	인터넷 뉴스 서비스

Web Container[Tomcat] 설치

I. JDK가 먼저 설치되어 있어야 함



The screenshot shows a web browser window with the address bar displaying `tomcat.apache.org`, which is highlighted with a red box. The browser's tab bar shows several open tabs, including '네이버', 'Google', '객체지향', 'The Java™ Tutorials', 'The Java EE 6 Tutorial', and 'json'. The main content area of the browser displays the Apache Tomcat website. On the left, there is a logo of a yellow cat with the text 'TM' and the words 'Apache Tomcat'. To the right of the cat logo is the text 'The Apache Software Foundation' with a colorful feather logo and the URL 'http://www.apache.org/'. Below this, there are two search boxes labeled 'Search the Site' and 'Search Site'. The main content area is divided into two columns. The left column has a heading 'Apache Tomcat' and a list of links: 'Home', 'Taglibs', 'Maven Plugin', and 'Download'. The 'Download' section is expanded, showing a list of links: 'Which version?', 'Tomcat 7.0', 'Tomcat 6.0', 'Tomcat 5.5', 'Tomcat Connectors', 'Tomcat Native', and 'Archives'. The 'Tomcat 7.0' link is highlighted with a red box. The right column has a heading 'Apache Tomcat' and two paragraphs of text. The first paragraph describes Apache Tomcat as an open source software implementation of the Java Servlet and JavaServer Pages technologies. The second paragraph describes the development environment and the Apache License version 2. The third paragraph describes the mission-critical web applications powered by Apache Tomcat.

tomcat.apache.org

네이버 Google 객체지향 The Java™ Tutorials The Java EE 6 Tutorial json

 **Apache Tomcat**

 **The Apache Software Foundation**
http://www.apache.org/

Search the Site Search Site

Apache Tomcat

- [Home](#)
- [Taglibs](#)
- [Maven Plugin](#)
- Download**
 - [Which version?](#)
 - [Tomcat 7.0](#)
 - [Tomcat 6.0](#)
 - [Tomcat 5.5](#)
 - [Tomcat Connectors](#)
 - [Tomcat Native](#)
 - [Archives](#)

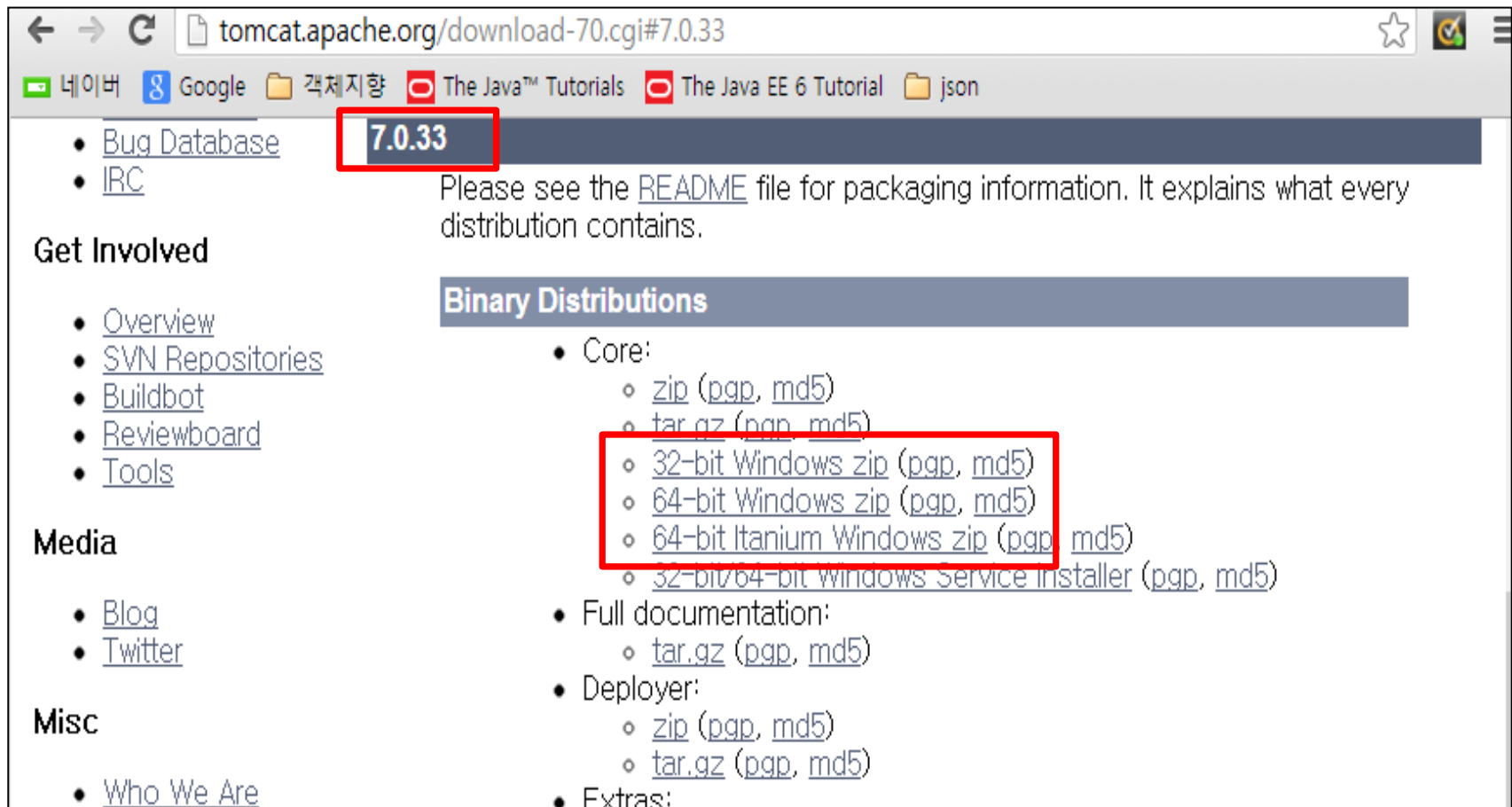
Apache Tomcat

Apache Tomcat is an open source software implementation of the Java Servlet and JavaServer Pages technologies. The Java Servlet and JavaServer Pages specifications are developed under the [Java Community Process](#).

Apache Tomcat is developed in an open and participatory environment and released under the [Apache License version 2](#). Apache Tomcat is intended to be a collaboration of the best-of-breed developers from around the world. We invite you to participate in this open development project. To learn more about getting involved, [click here](#).

Apache Tomcat powers numerous large-scale, mission-critical web applications across a diverse range of industries and organizations. Some of these users and their stories are listed on the [PoweredBy](#) wiki page.

Web Container[Tomcat] 설치



The screenshot shows the Apache Tomcat download page for version 7.0.33. The browser's address bar displays 'tomcat.apache.org/download-70.cgi#7.0.33'. The page features a sidebar with links to 'Bug Database', 'IRC', 'Get Involved' (including Overview, SVN Repositories, Buildbot, Reviewboard, and Tools), 'Media' (Blog, Twitter), and 'Misc' (Who We Are). The main content area highlights the version '7.0.33' and provides a link to the 'README' file. Under the 'Binary Distributions' section, a list of download options is shown, with '32-bit Windows zip (pgp, md5)', '64-bit Windows zip (pgp, md5)', and '64-bit Itanium Windows zip (pgp, md5)' highlighted by a red box. Other options include '32-bit/64-bit windows Service Installer (pgp, md5)', 'Full documentation' (tar.gz, pgp, md5), 'Deployer' (zip, tar.gz, pgp, md5), and 'Extras'.

tomcat.apache.org/download-70.cgi#7.0.33

7.0.33

Please see the [README](#) file for packaging information. It explains what every distribution contains.

Binary Distributions

- Core:
 - [zip \(pgp, md5\)](#)
 - [tar.gz \(pgp, md5\)](#)
 - [32-bit Windows zip \(pgp, md5\)](#)
 - [64-bit Windows zip \(pgp, md5\)](#)
 - [64-bit Itanium Windows zip \(pgp, md5\)](#)
 - [32-bit/64-bit windows Service Installer \(pgp, md5\)](#)
- Full documentation:
 - [tar.gz \(pgp, md5\)](#)
- Deployer:
 - [zip \(pgp, md5\)](#)
 - [tar.gz \(pgp, md5\)](#)
- Extras:

Tomcat PATH 설정[unused eclipse]

1. JAVA_HOME 환경 변수 생성

이름	값
JAVA_HOME	Jdk가 설치된 홈디렉토리

2. CATALINA_HOME 환경 변수 생성

이름	값
CATALINA_HOME	Tomcat이 설치된 홈 디렉토리

3. Path 환경 변수에 추가

이름	값
path	%JAVA_HOME%\bin;%CATALINA_HOME%\bin;



Servlet

학습내용

Http 요청 방식

Servlet 동작 원리

Servlet Programming

Page이동 방식

Session Trasking

JNDI와 DataSource를 이용한 Connection Pool

Http 요청 방식

HTTP 요청 및 응답 구조



HTTP 요청 방식

I. Query String 전송방식

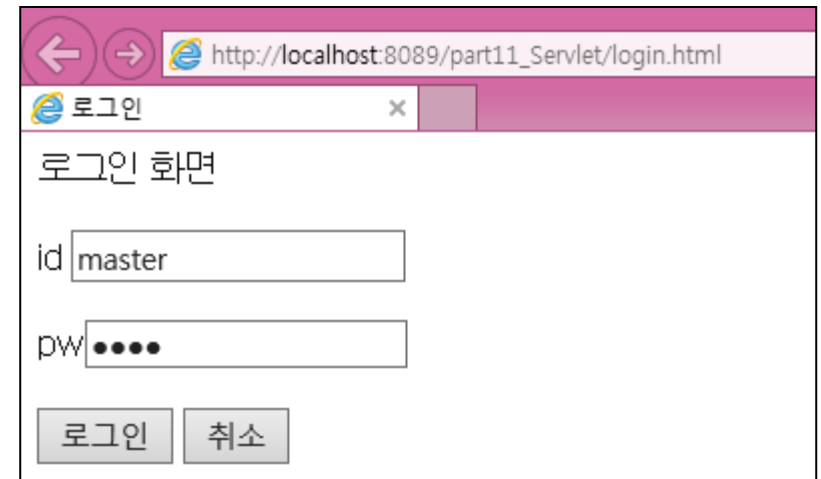


HTTP Get방식 요청

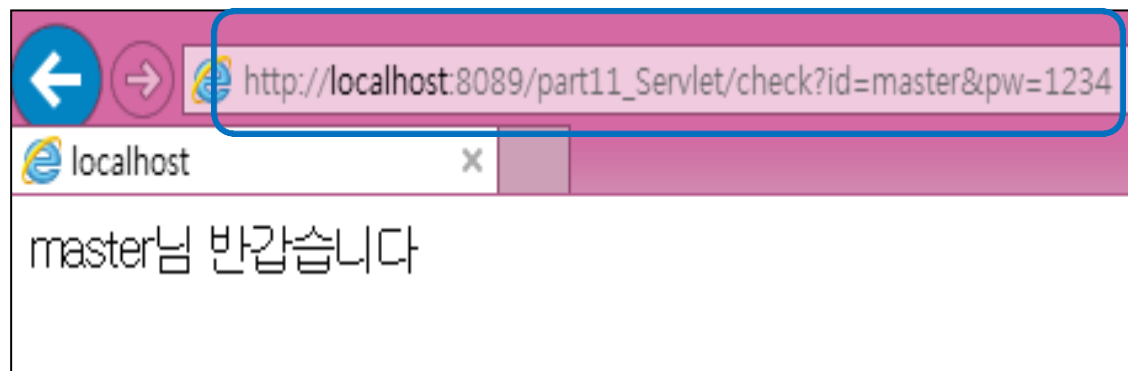
html form

로그인 화면<p>

```
<form action="check" method="get">  
  id <input type="text" name="id"><p>  
  pw <input type="password" name="pw"><p>  
  <input type="submit" value="로그인">  
  <input type="reset" value="취소">  
</form>
```



A screenshot of a web browser window. The address bar shows the URL `http://localhost:8089/part11_Servlet/login.html`. The page title is "로그인" (Login). The content area displays the text "로그인 화면" (Login screen). Below this, there are two input fields: "id" with the value "master" and "pw" with masked characters "....". At the bottom, there are two buttons: "로그인" (Login) and "취소" (Cancel).



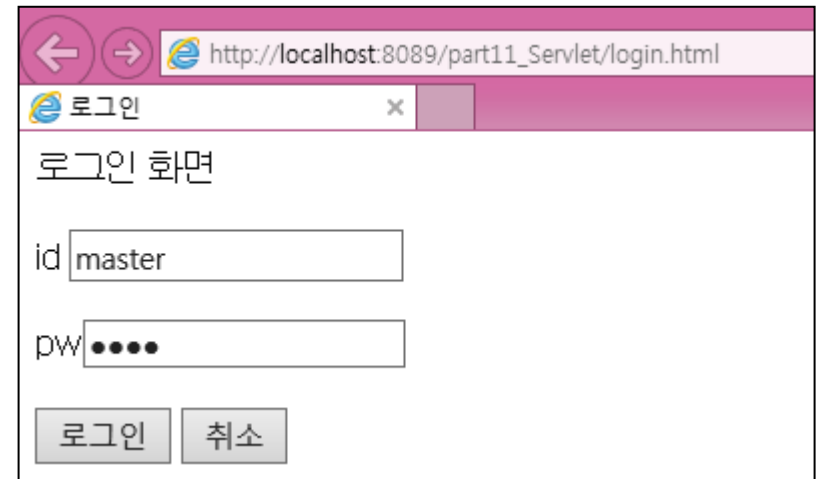
A screenshot of a web browser window. The address bar shows the URL `http://localhost:8089/part11_Servlet/check?id=master&pw=1234`. The page title is "localhost". The content area displays the text "master님 반갑습니다" (Welcome, master).

HTTP POST 방식

html form

로그인 화면 <p>

```
<form action="check" method="post">
  id <input type="text" name="id"><p>
  pw <input type="password" name="pw"><p>
  <input type="submit" value="로그인">
  <input type="reset" value="취소">
</form>
```

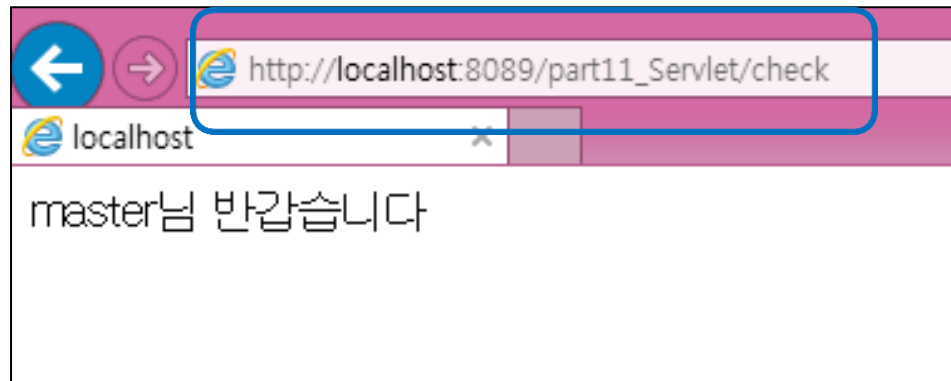


로그인 화면

id master

pw

로그인 취소

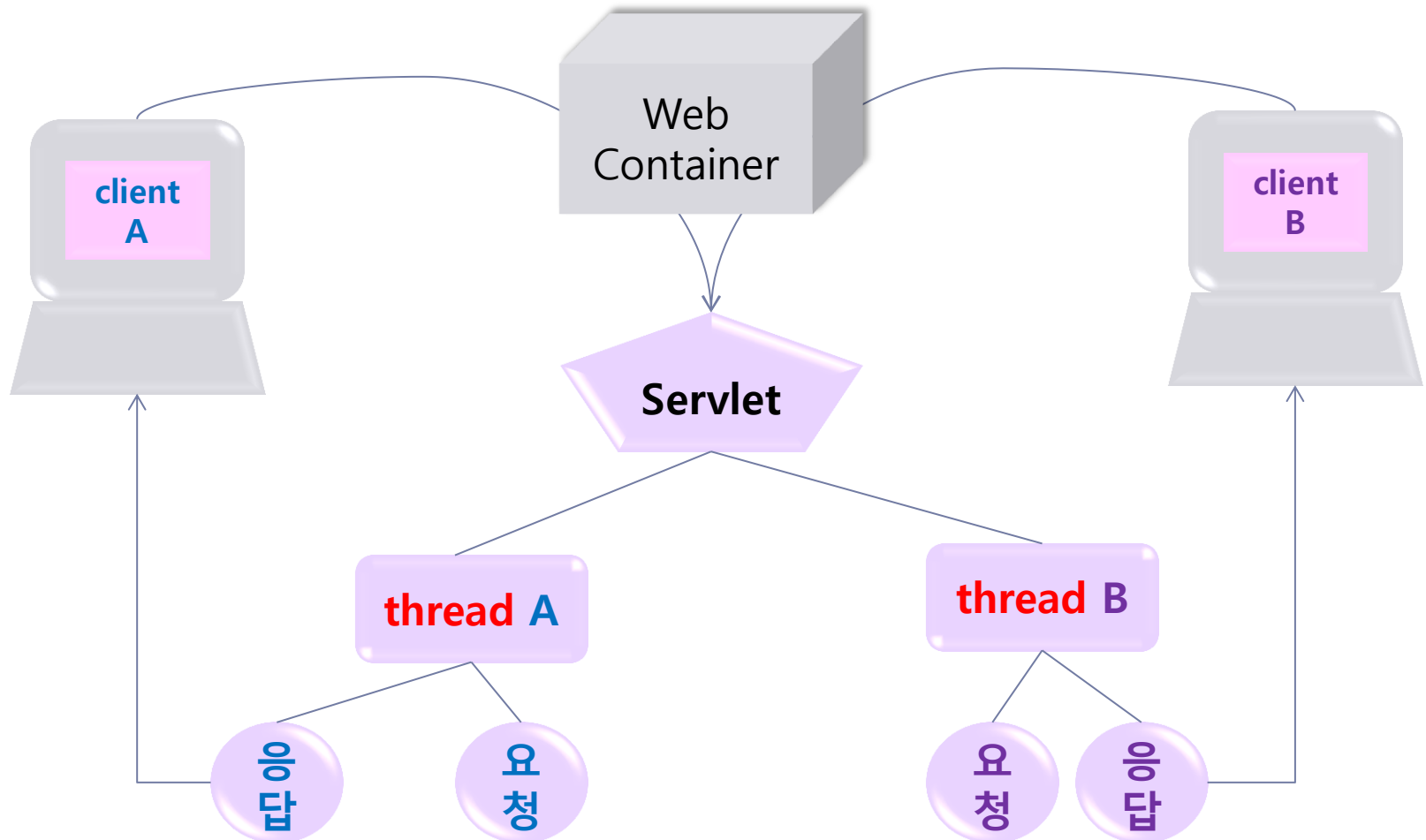


master님 반갑습니다

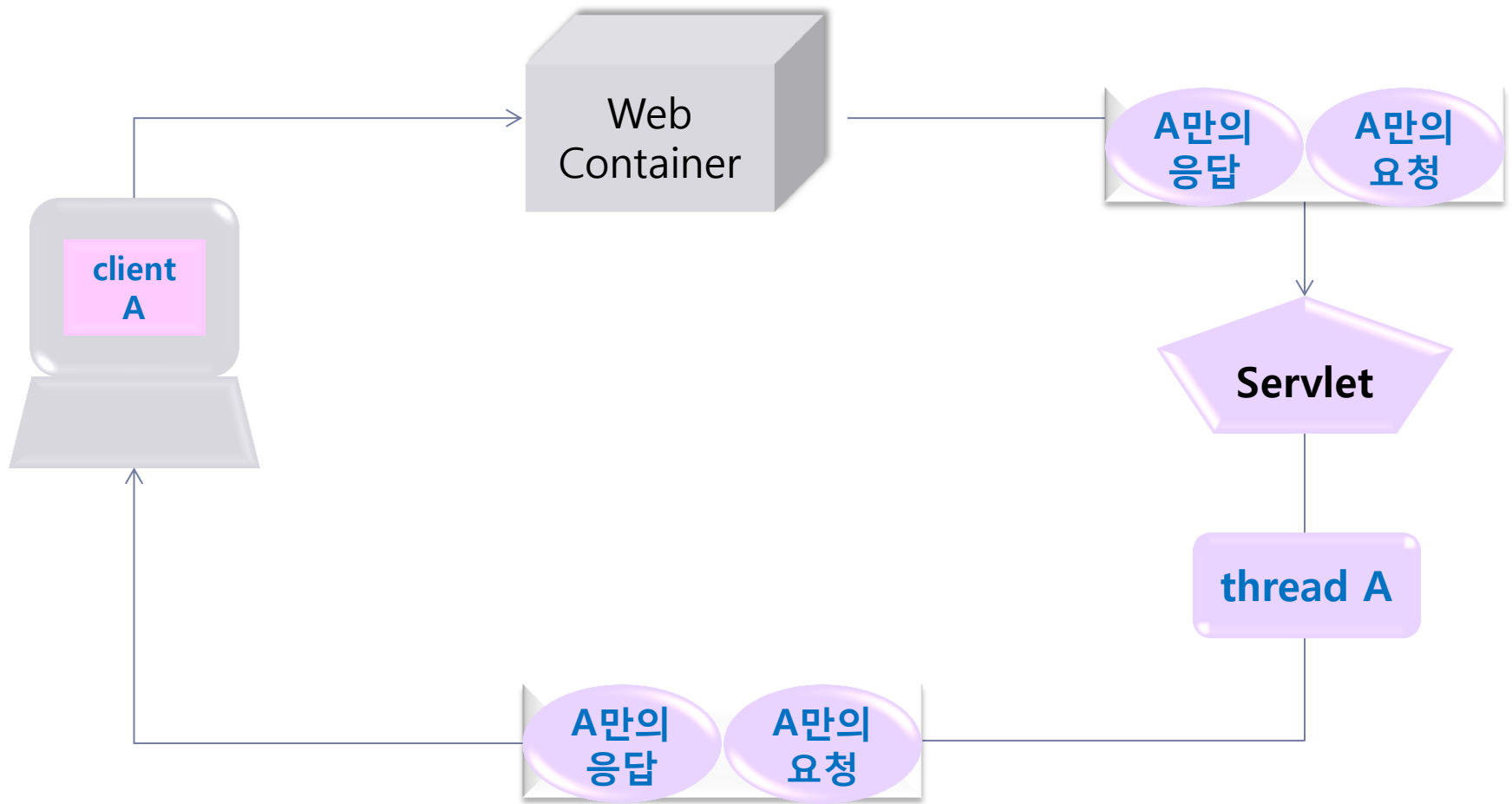
Servlet 동작 원리

Servlet 동작 원리 – Client 요청 처리

1. Client 요청은 개별 thread로 생성되어 실행됨

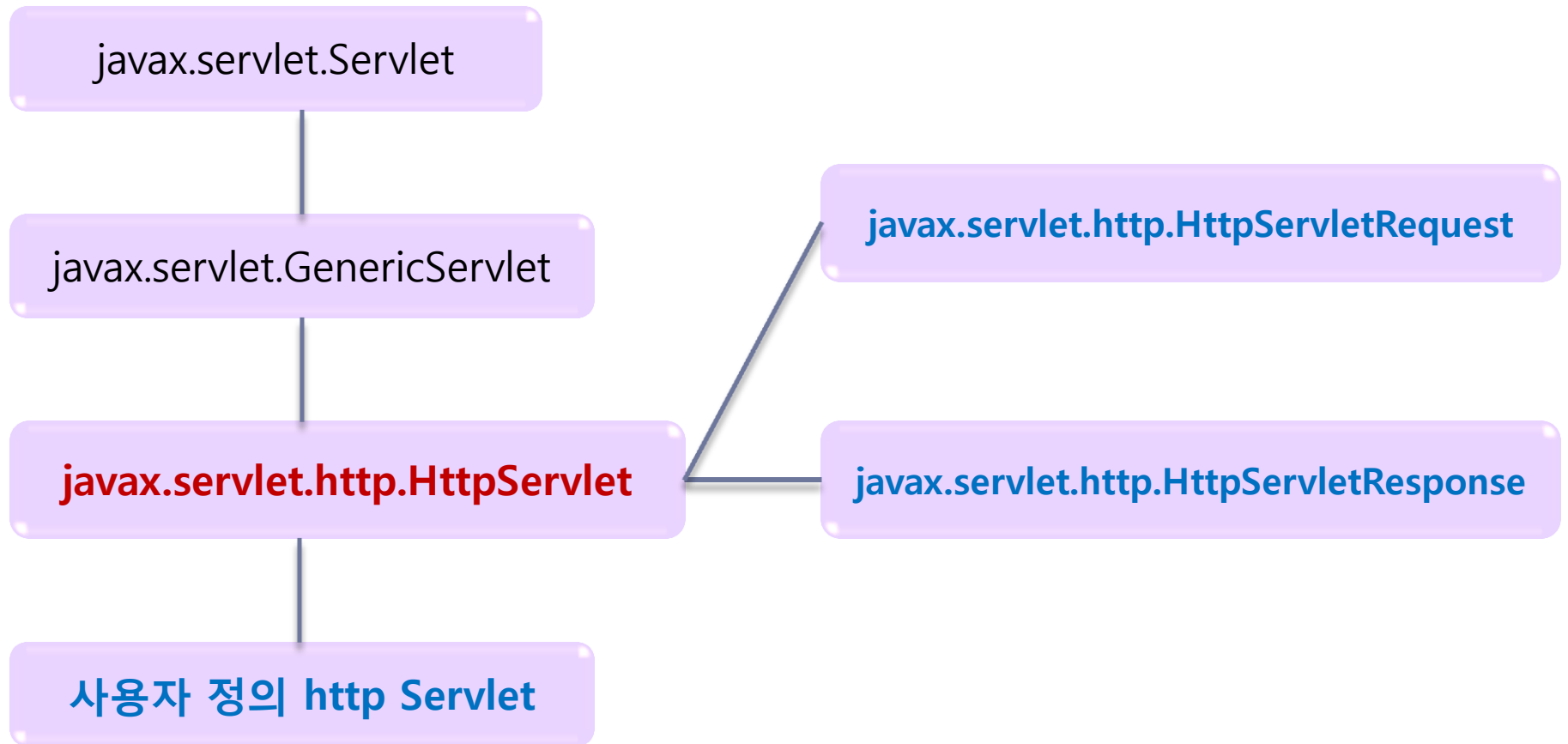


Servlet 실행 Process



Servlet Programming

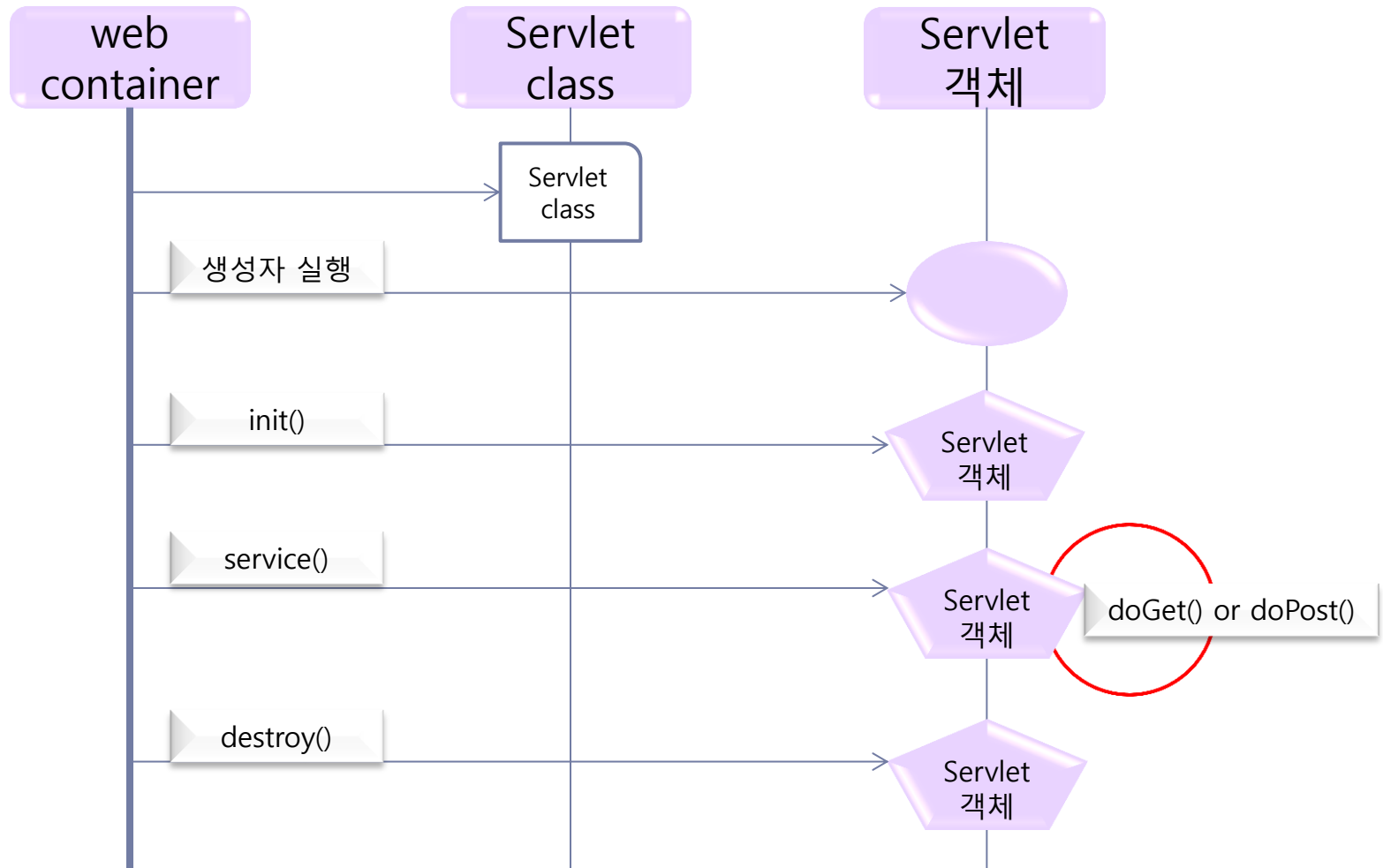
Servlet 주요 API



Servlet 개발 방법

```
3 import java.io.IOException;
4 import java.io.PrintWriter;
5
6 import javax.servlet.ServletException;
7 import javax.servlet.http.HttpServlet;
8 import javax.servlet.http.HttpServletRequest;
9 import javax.servlet.http.HttpServletResponse;
10
11 public class LoginCheck extends HttpServlet {
12     public LoginCheck() {}
13
14     //get방식 요청 처리 메소드
15     protected void doGet(HttpServletRequest request, HttpServletResponse response)
16         throws ServletException, IOException {
17         process(request, response);
18     }
19
20     //post방식 요청 처리 메소드
21     protected void doPost(HttpServletRequest request, HttpServletResponse response)
22         throws ServletException, IOException {
23         process(request, response);
24     }
25
26     protected void process(HttpServletRequest request, HttpServletResponse response)
27         throws ServletException, IOException {
28         response.setContentType("text/html;charset=euc-kr");
29         PrintWriter out = response.getWriter();
30         out.println(request.getParameter("id") + "님 반갑습니다");
31         out.close();
32     }
33 }
```

Servlet Life Cycle



Servlet Life Cycle 주요 API

init()

1. 컨테이너가 Servlet 객체 생성한 후에 1번 호출
2. 서블릿을 초기화 로직으로 재정의

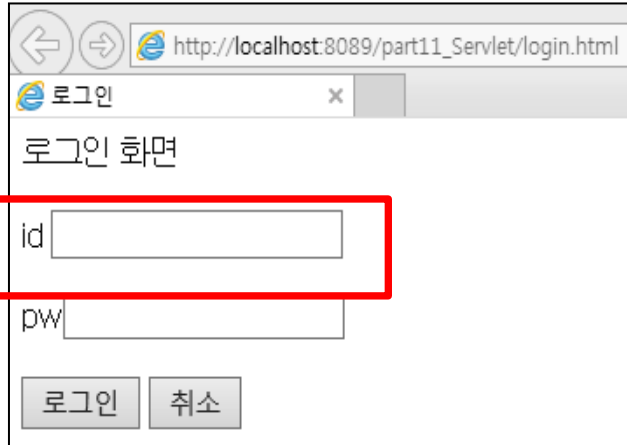
service()

1. client의 요청 발생 시 컨테이너에서 스레드를 이용하여 호출
2. client의 요청별 매번 호출
3. http 요청 방식별 `doGet()`, `doPost()` 등으로 분기시킴

destroy()

1. 컨테이너가 servlet 객체를 메모리에서 unload하기 직전에 1번 호출
2. 사용 자원 반환 로직으로 주로 재정의

Form Parameter 처리 process



로그인 화면

id

pw

로그인 취소

```
14 //get방식 요청 처리 메소드
15 protected void doGet(HttpServletRequest request, HttpServletResponse response)
16     throws ServletException, IOException {
17     process(request, response);
18 }
19 //post방식 요청 처리 메소드
20 protected void doPost(HttpServletRequest request, HttpServletResponse response)
21     throws ServletException, IOException {
22     process(request, response);
23 }
24 protected void process(HttpServletRequest request, HttpServletResponse response)
25     throws ServletException, IOException {
26     response.setContentType("text/html;charset=utf-8");
27     PrintWriter out = response.getWriter();
28
29     String id = request.getParameter("id");
30     String pw = request.getParameter("pw");
31
```

```
<form action="check" method="post">
```

```
id <input type="text" name="id"><p>
```

```
pw <input type="password" name="pw"><p>
```

```
<input type="submit" value="로그인">
```

```
</form>
```

```
id.equals("master") && pw.equals("7777")){
```

```
out.println("<html>");
```

```
out.println("<body>");
```

```
out.println(id + "님 반갑습니다");
```

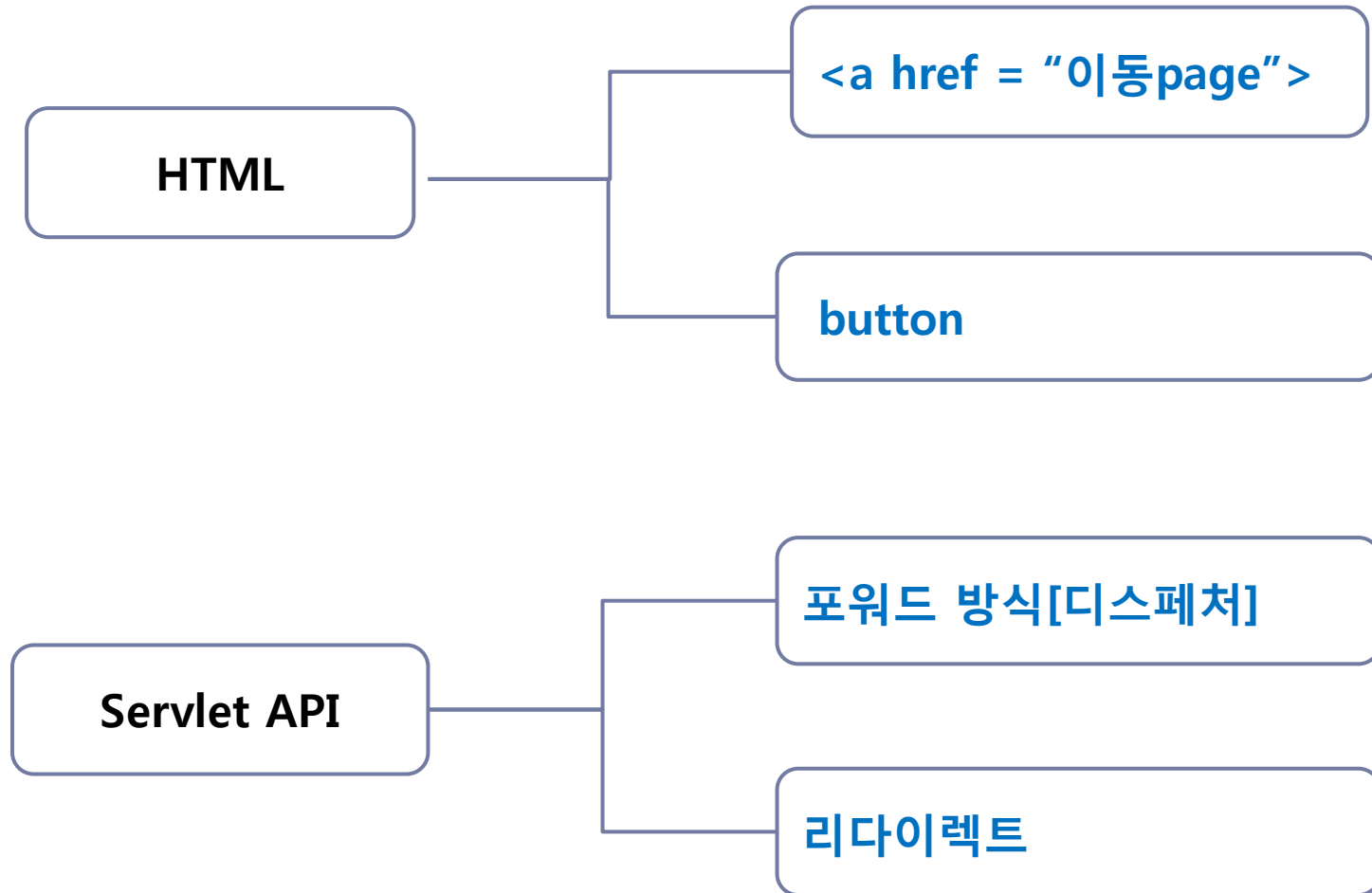
```
out.println("</body>");
```

```
out.println("</html>");
```

Check Box와 같은 다중 데이터 =
String [] values = request.getParameterValues("checkbox form tag");

Page이동 방식

Web Page 이동



Step01_basic에서 실습한 내용

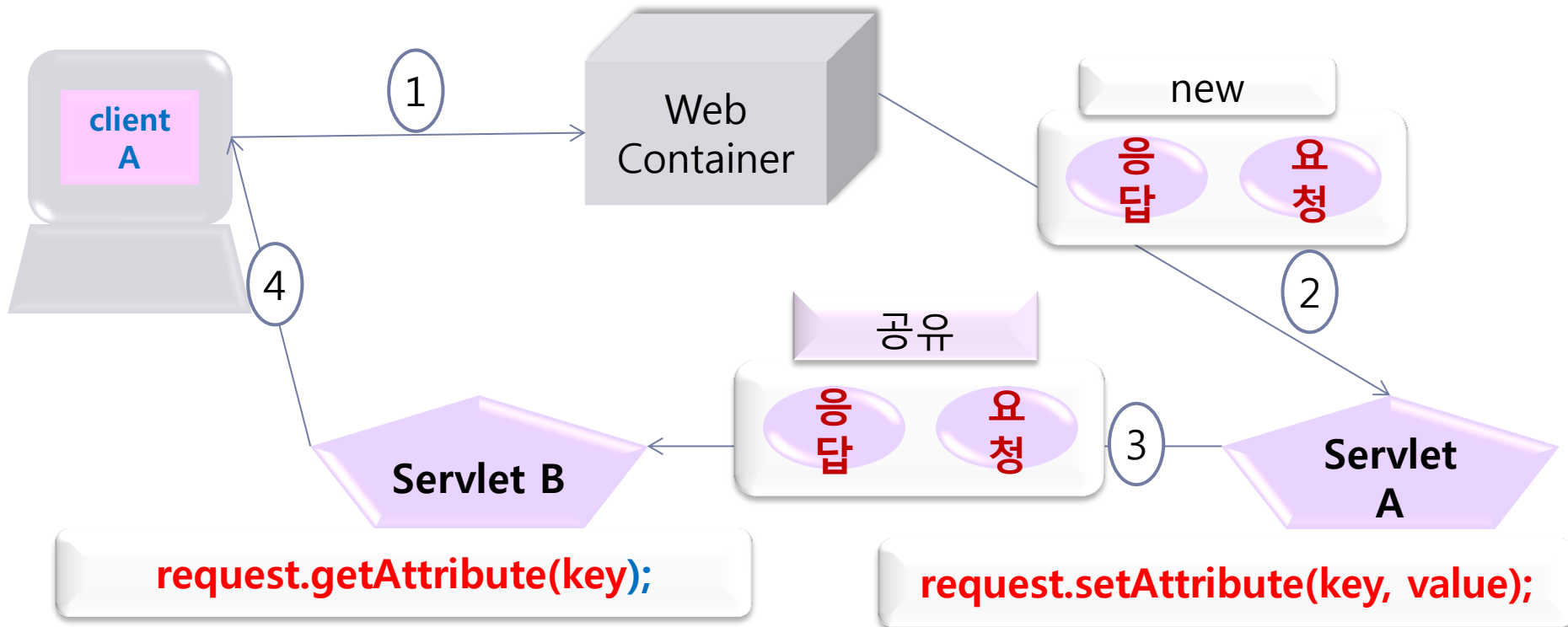
성공 : 포워드

1. login.html -> LoginCheck -> Success
2. http://localhost/step01_basic/login.html
3. http://localhost/step01_basic/kdata2?idValue=master&pwValue=w
4. login.html : post방식 -> LoginCheck : doPost() -> Success : doPost()
5. login.html : get방식 -> LoginCheck : doGet() -> Success : doGet()

실패 : 리다이렉트 - 무조건 get방식으로 인식

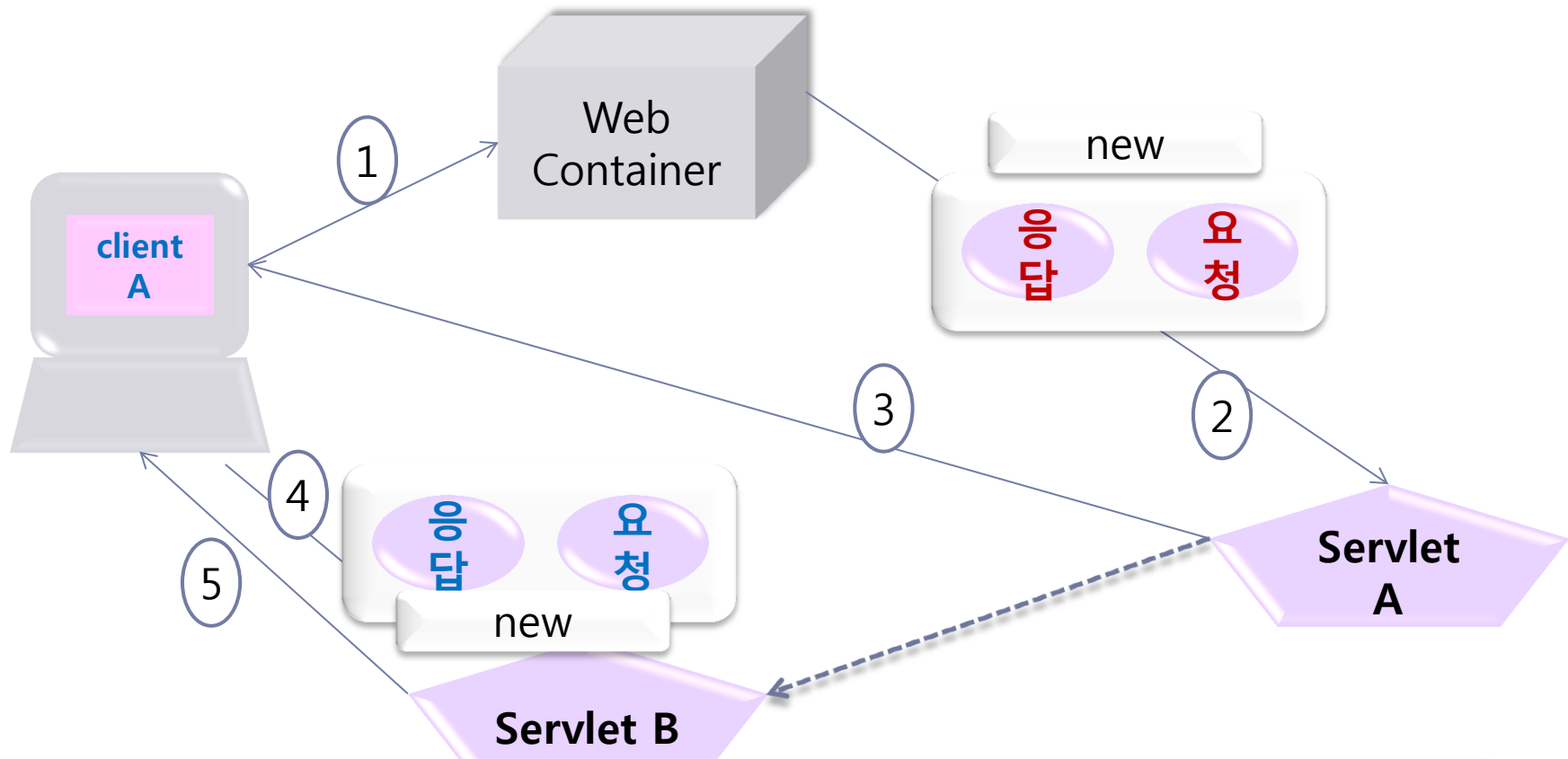
1. login.html -> LoginCheck -> Fail
2. http://localhost/step01_basic/login.html
3. http://localhost/step01_basic/fail
4. login.html : post방식 -> LoginCheck : doPost() -> Fail : doGet()
5. login.html : get방식 -> LoginCheck : doGet() -> Fail : doGet()

Servlet API를 활용한 Page이동 – forward방식



```
request.getRequestDispatcher("이동page").forward(request, response);
```

Servlet API를 활용한 Page이동 – **redirect** 방식



```
response.sendRedirect("이동page");
```

Session Tracking

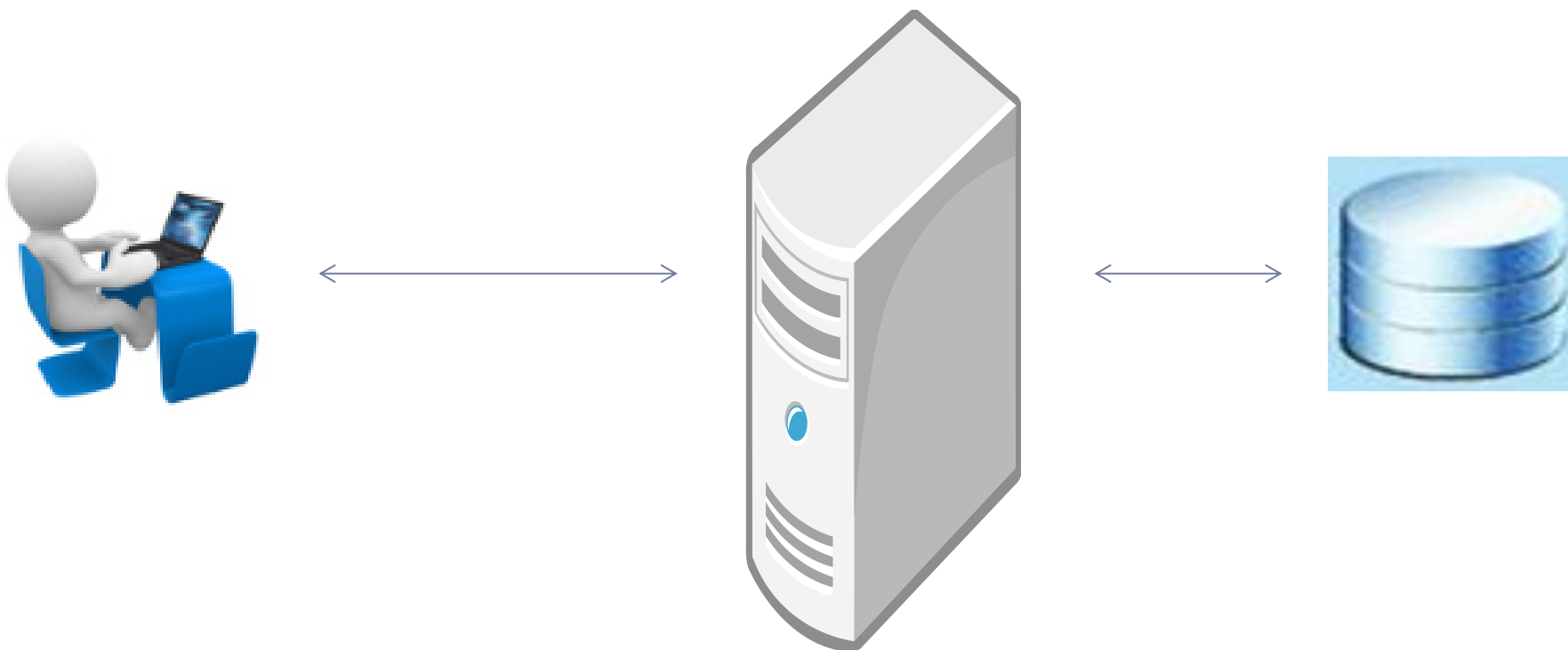
- 세션
- 쿠키

Page이동 방식과 무관하게 일관성 있는 데이터 유지 기술

1. 세션 : Client의 상태 유지를 위한 데이터를 서버 시스템에 저장
2. 쿠키 : Client의 상태 유지를 위한 데이터를 client 시스템에 저장



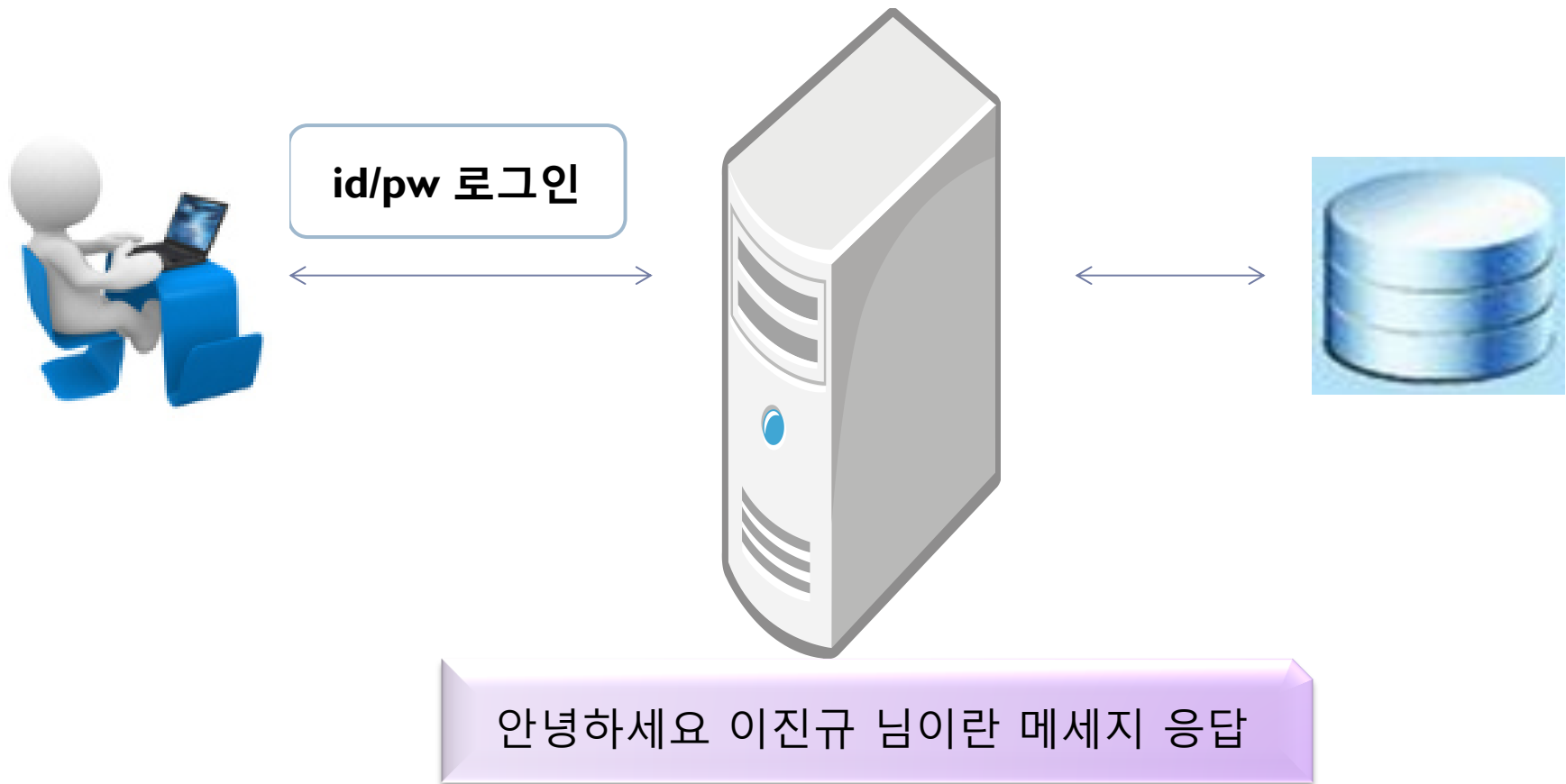
생각하기 – http protocol 특징



로그인 하기전 user가 누구인지 구분 불가



생각하기 – http protocol 특징



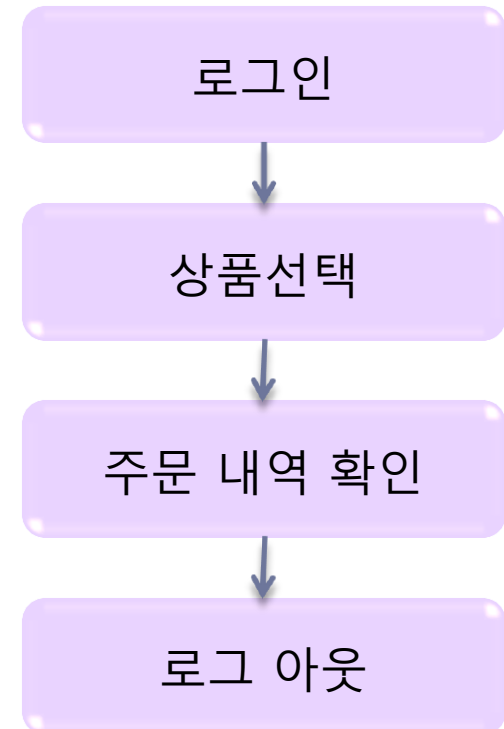
Session Tracking

1. 등장배경

1. HTTP 프로토콜 - 비 연결 지향형

2. Session Tracking

1. 일정 시간동안 동일한 사용자의 여러 요청들을 유지 관리하기 위한 기술
1. 웹 애플리케이션과 웹 브라우저 사이에서의 정보 교환



Session Tracking 적용 방법

Cookie

상태정보를 Client System
에(파일/브라우저메모리)
저장하여 유지하는 기술

Session

상태정보를 Server System
에 저장하여 유지하는 기술

Cookie 구조



Cookie Application 개발

Cookie 생성 및 데이터 저장

```
String id = request.getParameter("id");
String pw = request.getParameter("pw");

if(id.equals("master") && pw.equals("7777")){
    Cookie idCookie = new Cookie("id", id);
    idCookie.setMaxAge(60*60*24);
    response.addCookie(idCookie);
    response.sendRedirect("success");
}else{
    request.setAttribute("errorMsg", "id/pw 오류");
    request.getRequestDispatcher("fail").forward(request, response);
}
```

Cookie 획득 및 데이터 획득

```
Cookie [] allCookie = request.getCookies();
for(int index = 0; index < allCookie.length; index++){
    if( allCookie[index].getName().equals("id")){
        id = allCookie[index].getValue();
    }
}
```

Session 구조

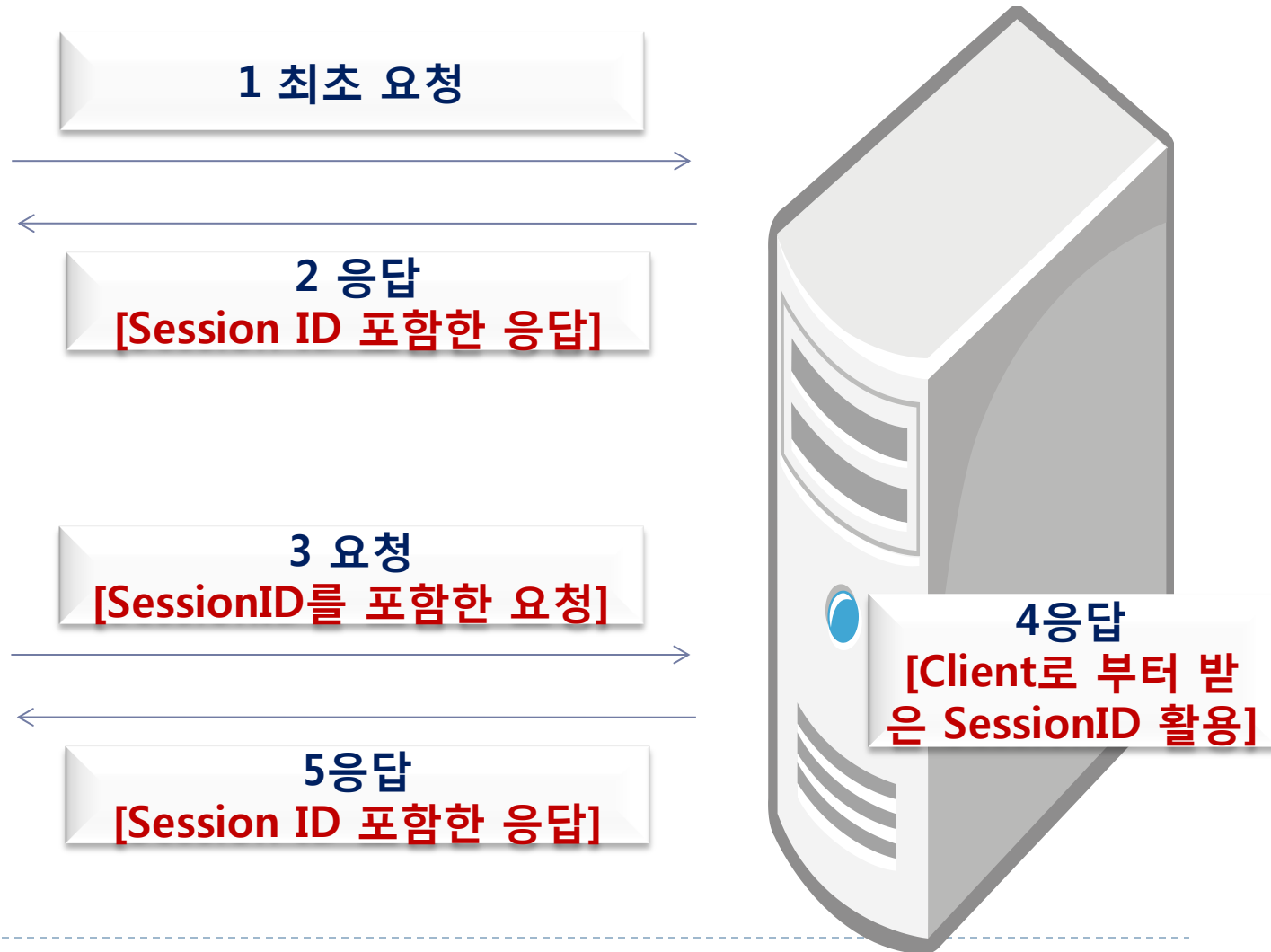


서버가 관리하는 세션

세션 ID/ID/로그인상태/
주문내역/주문완료유무....

세션 ID : 웹 서버에 요청한 Client를 식별할 수 있는 데이터

Session 구조



Session Application 개발

HttpSession 생성 및 데이터 저장

```
30
31     String id = request.getParameter("id");
32     String pw = request.getParameter("pw");
33
34     if(id.equals("master") && pw.equals("7777")){
35
36         HttpSession session = request.getSession();
37         session.setAttribute("id", id);
38         session.setAttribute("pw", pw);
39
39         response.sendRedirect("success");
40     }else{
41         request.setAttribute("errorMsg", "id/pw 오류");
42         request.getRequestDispatcher("fail").forward(request, response);
43     }
44
45 }
```

HttpSession 획득 및 데이터 획득

```
HttpSession session = request.getSession();
String id = (String)session.getAttribute("id");
```

ServletContext API

1. 웹 어플리케이션마다 1개씩 만들어 짐
2. 서버가 웹 어플리케이션을 시작시킬 때 생성 함
3. 웹 어플리케이션 내의 모든 Servlet과 JSP에서 공유
 1. `setAttribute()` & `getAttribute()`
4. `web.xml`의 설정을 통해 ServletContext 객체에 초기화 파라미터 전달 가능(여러 개 가능)
5. ServletContext를 통해 초기화 파라미터 조회 가능

ServletContext(2/3)

```
6
7 <!-- 중략 -->
8 <context-param>
9   <param-name>companyName</param-name>
10  <param-value>my.com</param-value>
11 </context-param>
12 <!-- 중략 -->
13
```

web.xml

1. Web Application이 초기화 될때, 단 한번만 초기화 파라미터를 설정파일에서 읽어 들임
 1. 서버가 web.xml을 읽어 들임
 2. 서버가 ServletContext 객체를 생성함
 3. 서버는 컨텍스트 초기화 파라미터의 이름/값 쌍을 만듦
 1. 여러개 설정 되어 있는 경우 파라미터 설정 수만큼 반복
 4. 서버는 생성한 컨텍스트 초기화 파라미터의 String쌍을 ServletContext 객체에 저장함

ServletContext(3/3)

1. void getInitParameter(String name)

1. 초기화 파라미터 조회기능

```
1 package com.my.main;
2
3 import javax.servlet.http.*;
4
5
6
7
8 public class MainServlet extends HttpServlet
9 {
10     public void doGet(HttpServletRequest request,
11                        HttpServletResponse response)
12         throws ServletException, IOException
13     {
14         ServletContext ctx = getServletContext();
15         String companyName = ctx.getInitParameter("companyName");
16
17         response.setContentType("text/html;charset=utf-8");
18         PrintWriter out = response.getWriter();
19         out.println("<HTML>");
20         out.println("<BODY>");
21         out.println("<h1>메인화면 입니다.</h1>");
22         out.println("<hr>");
23         out.println("<font color='blue'><h2>"+companyName+"</h2></font>");
24         out.println("</BODY>");
25         out.println("</HTML>");
26     }
27 }
```

메인화면 입니다.

my.com

데이터 저장 및 활용을 위한 주요 API

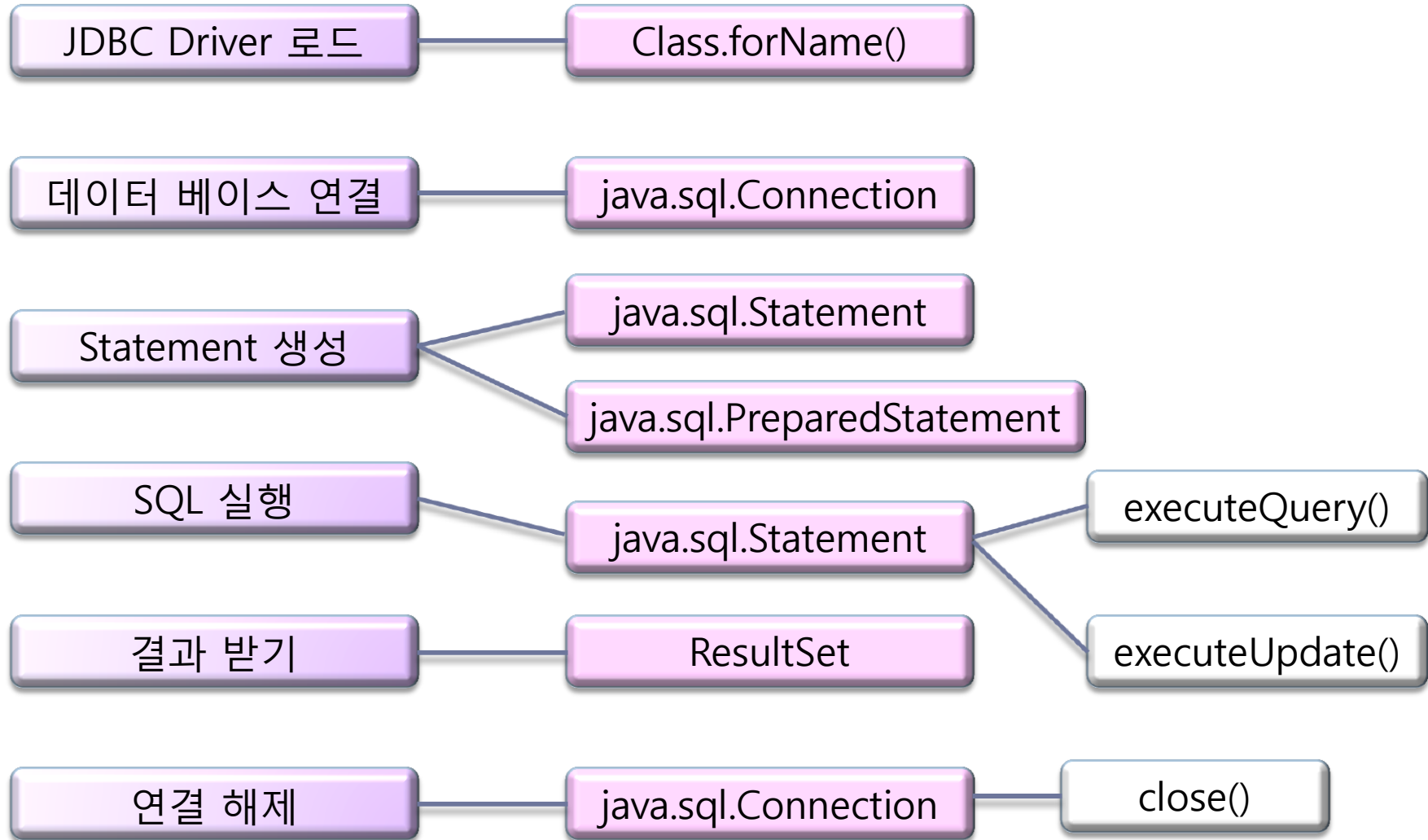
API		
HttpServletRequest - Client가 접속시 자동 생성	setAttribute(key, value)	getAttribute(key)
HttpSession - getSession()	setAttribute(key, value)	getAttribute(key)
ServletContext	setAttribute(key, value)	getAttribute(key)

Forward 방식으로 이동되는 경우에 한해서만 유효한 범위

Page 이동 방식과 무관하게 세션이 존재하는한 데이터 공유 가능

JNDI와 DataSource를 이용한 Connection Pool

JDBC API를 활용한 DB연동 개발 단계



IT인들이 가장 두려워 하는 것

1 순위 - 데이터 손실

2 순위 - 서버 시스템 다운

3 순위 - 빈번한 요구사항...

Connection Pool의 개요(1/2)

1. 등장배경

1. DB작업마다 Connection 생성에 많은 시간과 비용
2. 애플리케이션에서 새로운 DB 연결을 만드는 것은 시스템 부하의 한 요소
3. 사용하지 않는 커넥션으로 인한 데이터베이스 자원낭비
4. 대규모 시스템일수록 커넥션 관리는 중요

2. Connection Pool(CP)

1. 일정 수 만큼의 Connection을 pool로 유지
2. 일정수준 이상의 커넥션을 유지하기 때문에 최적의 성능을 보장
 1. DB 서버 시스템 다운 방지
3. 사용하지 않는 커넥션의 자동관리 기능
4. 최대 접속에 대한 제한 기능
5. 여러 애플리케이션에 서비스 가능

Connection Pool의 개요(2/2)

1. Connection Pool 관리방법

1. 애플리케이션 서버가 startup 될 때 일정수의 커넥션을 미리 생성
2. 애플리케이션의 요청에 따라 생성된 커넥션 객체를 전달
3. 일정 수 이상의 커넥션이 사용되어 지면 새로운 커넥션을 생성
4. 사용되지 않는 커넥션은 정리하고 최소수의 커넥션을 유지

2. 예전에는 별도로 구현된 Connection Pool 사용

3. JDBC 2.0 이상 - javax.sql.DataSource 인터페이스 제공

4. 애플리케이션 서버에서 제공하는 DataSource 구현체를 권장함

DataSource(1/3)

1. JNDI(Java Naming and Directory Interface)
 1. 표준화된 디렉토리 및 이름 서비스에 대한 자바 인터페이스
 2. JNDI를 통해 객체 참조가 가능함
 3. 애플리케이션에서 DataSource를 JNDI를 통해 제공

DataSource(2/3)

Property	Value	설명
JNDI Name	jdbc/oraDB	네이밍 서비스에서 사용할 이름을 입력 일반적으로 DataSource에는 jdbc 접두어 사용
Data Source URL	jdbc:oracle:thin:@xxx.xx.xxx.xxx:1521:OSID	데이터베이스 접속을 위한 URL 정보를 입력 각 데이터베이스별 정의된 URL을 입력
JDBC Driver Class	oracle.jdbc.OracleDriver	JDBC 클래스 로딩을 위한 클래스 정보
User Name	scott	데이터베이스 계정
Password	tiger	데이터베이스 계정의 비밀번호
Max. Active Connections	4	커넥션 풀에서 관리할 최대 동시 연결 수 지정 데이터베이스 사용규모에 따라 적절히 설정 커넥션풀은 Max 값 이상의 연결은 만들지 못함
Max Idle Connections	2	최대로 유지할 유휴 연결의 개수 설정값 이상의 커넥션은 자동으로 정리 →항상유지할 빈 연결 개수로 생각할 수 있음
Max Wait for Connection	50000	밀리세컨드 값(기본값은 50 초) 데이터베이스 연결을 구하기 위한 대기 시간 →시간이 지나치게 짧으면 클라이언트에 예외를 발생시킬 수 있음.

DataSource(3/3)

```
1 <?xml version="1.0" encoding="UTF-8"?>
2
3 <Context>
4
5   <WatchedResource>WEB-INF/web.xml</WatchedResource>
6
7   <Resource
8     auth="Container"
9     driverClassName="oracle.jdbc.OracleDriver"
10    maxActive="5"
11    maxIdle="3"
12    maxWait="-1"
13    name="jdbc/oraDB"
14    password="oracle"
15    type="javax.sql.DataSource"
16    url="jdbc:oracle:thin:@125.128.26.240:1521:ORCL"
17    username="user01" />
18
19 </Context>
```

```
3 import java.sql.*;
4 import javax.sql.*;
5 import javax.naming.*;
6
7 public class UserDao {
8     DataSource ds = null;
9     public UserDao() {
10         try {
11             // JNDI 이용하여 DataSource 찾을
12             InitialContext ctx = new InitialContext();
13             ds = (DataSource) ctx.lookup("java:comp/env/jdbc/oraDB");
14         } catch (Exception e) {
15             e.printStackTrace();
16         }
17     }
18     public User findUser(String userId) {
19         Connection con = null;
20         PreparedStatement pstmt = null;
21         ResultSet rset = null;
22         User user = null;
23         String sql = "SELECT * FROM userinfo where userid = ?";
24         try {
25             // DataSource에서 connection 얻음
26             con = ds.getConnection();
27             pstmt = con.prepareStatement(sql);
28             pstmt.setString(1, userId);
```



JSP

Java Server Page

학습내용

JSP 개요 및 동작 원리

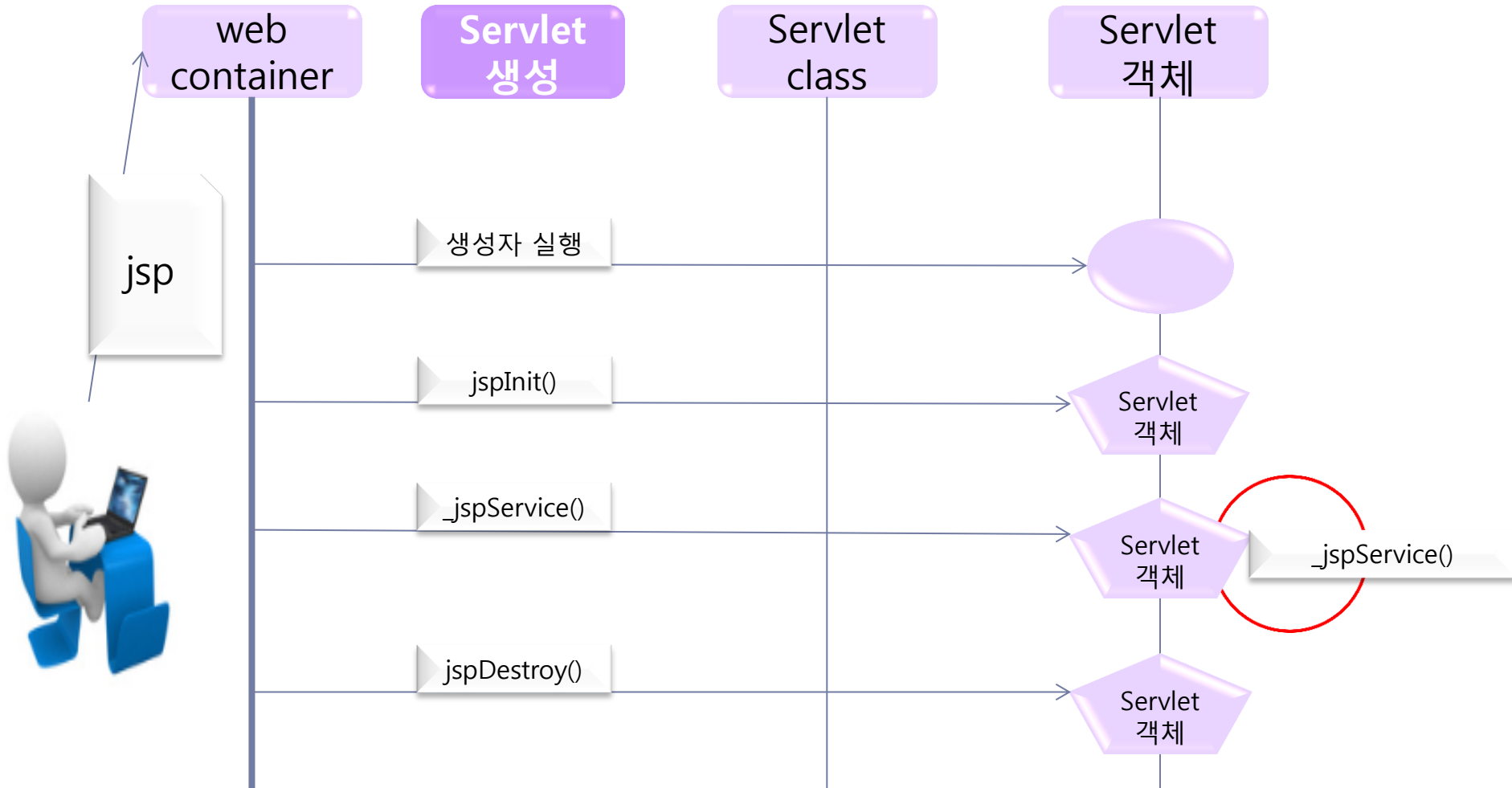
JSP Tag

JSP 개요 및 동작 원리

Java Web Application[Servlet & JSP]

	Servlet	JSP
공통점	<ol style="list-style-type: none"> 1. 웹 콘텐츠를 생성하기 위한 Java 2. Web Container 내에서 실행 3. client 요청을 thread로 처리 	<p>현추세 : 가급적 jsp는 자바코드 최소화 권장 EL/JSTL</p>
차이점	<ol style="list-style-type: none"> 1. 확장자 - *.java 2. 구성 : 자바 문법 + 경우에 따라 HTML tag 3. Controller 로직 권장 	<ol style="list-style-type: none"> 1. 확장자 - *.jsp 2. 구성 : 자바코드 + HTML tag + JSP Scripting tag + JSP Action tag + EL + JSTL 3. 실행시 Web Container가 Servlet으로 자동 변환 4. View 로직 권장 5. 내장 객체 자동 생성 [request, response, session...]

JSP 동작 원리 및 life Cycle



JSP 내장객체 - 개요

1. JSP에서 선언하지 않고 사용할 수 있는 객체
2. 컨테이너에 의해 변환되어지는 서블릿 클래스의 service() 구현부내에 선언되어지는 참조 변수
3. 사용 가능 tag
 1. `<% %>` & `<%= %>` tag에서 사용 가능

JSP 내장객체 - 종류

참조 변수명(내장 객체)	자바 클래스와 주요 역할
request	<code>javax.servlet.http.HttpServletRequest</code> HTML Form 요소 선택 값과 같은 사용자 입력 정보를 읽어올 때 사용
response	<code>javax.servlet.http.HttpServletResponse</code> 사용자 요청에 대한 응답을 처리할 때 사용
pageContext	<code>javax.servlet.jsp.PageContext</code> 현재 JSP 실행에 대한 context 정보를 참조하기 위해 사용
session	<code>javax.servlet.http.HttpSession</code> 클라이언트 세션 정보를 처리하기 위해 사용
application	<code>javax.servlet.ServletContext</code> 웹 서버의 애플리케이션 처리와 관련된 정보를 참조하기 위해 사용
out	<code>javax.servlet.jsp.JspWriter</code> 사용자에게 전달하기 위한 output 스트림 처리하기 위해 사용
config	<code>javax.servlet.ServletConfig</code> 현재 JSP에 대한 초기화 환경을 처리하기 위해 사용
page	<code>java.lang.Object type</code> 현재 JSP 페이지에 대한 클래스 정보
exception	<code>ava.lang.Throwable</code> 예외 처리를 위해 사용

JSP Tag

- JSP Scripting Tag**
- JSP Action Tag**
- Expression Language**
- JSTL**

JSP Scripting Tag

JSP 주요 Scripting Tag

종 류	tag 명	특 징
<% %>	Scriptlet	순수 자바 코드 구현 service() 구현부
<%! %>	Declaration	멤버 변수 & 메소드 구현
<%@ %>	Directive	1. page directive : 인코딩 및 import 적용구등 <%@ page import="package.class" contentType="text/html;charset=utf-8" %> 2. include directive : 외부 file include <%@ include file="외부 파일" %> 3. tag directive : 외부 library 사용을 위한 선언 <%@ taglib prefix="이름" uri="고유 uri" %>
<%= %>	Expression	출력 tag, service() 구현부
<%-- --%>	comment	주석, 브라우저에 은닉

JSP의 기본 구성 요소 - Page Directive Tag 속성

속 성	설 명	기본값
language	스크립트 언어를 지정	java
import	jsp 파일 내에서 사용할 외부 자바 패키지나 클래스 지정	
session	세션 생성 여부 지정	true
buffer	버퍼 크기 지정	8kb
autoFlush	버퍼 내용 자동 비움 지정	true
isThreadSafe	단일 스레드 모델을 사용하여 동시성 제어 여부 지정	true
info	JSP 페이지 설명	
errorPage	에러 발생 시 호출 페이지 지정	
isErrorPage	에러만 처리하는 페이지 지정	false
contentType	MIME 형식과 캐릭터셋 설정	text/html; charset =ISO-8859-1

JSP Scripting Tag 예제

```
1 <%@ page language="java" contentType="text/html; charset=EUC-KR" pageEncoding="EUC-KR"%>
2 <!-- 1. page Directive --%>
3 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
4 <html>
5 <head>
6 <meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
7 <title>JSP Scripting Tag 학습</title>
8 </head>
9 <body>
10
11 <h3>JSP Scripting Tag 예제</h3><p><hr>
12
13 2. Declaration tag<br>
14 <%! String varName = "멤버 변수"; %>
15 <br><br>
16
17 3. Scriptlet tag<br>
18 <% out.println("client 요청시마다 실행되는 tag"); %>
19 <br><br>
20
21 4. Expression tag<br>
22 <%= "출력 tag" %>
23 <br><br>
24
25 5. Comment tag<br>
26 <!-- client 브라우저의 소스 보기 불가능한 주석 --%>
27
28 </body>
29 </html>
```

JSP Scripting Tag 예제

2. Declaration tag

3. Scriptlet tag

client 요청시마다 실행되는 tag

4. Expression tag

출력 tag

5. Comment tag

JSP Action Tag

JSP 주요한 Action Tag

종 류	특 징
<jsp:useBean>	자바 빈 객체 생성 <jsp:useBean id="이름" class="클래스명" scope="page request session application" />
<jsp:setProperty>	parameter값을 자바빈 객체의 멤버에 대입 <jsp:setProperty name="id명" property="property명" />
<jsp:forward>	포워드 방식으로 페이지 이동 <jsp:forward page="이동page" />
<jsp:include>	동적으로 외부 파일 include <jsp:include page="include할 page" />
<jsp:param>	forward 및 include시에 사용될 parameter값 적용 <jsp:param name="이름" value="값" />

Action tag - <jsp:useBean> 의 scope 속성

1. <jsp:useBean id="변수명" class="빈즈클래스명" scope="page | request | session | application" />
2. scope

scope	Servlet API
page	this
request	HttpServletRequest
session	HttpSession
application	ServletContext

JSP Action Tag 예제

- <jsp:useBean>, <jsp:setProperty>, <jsp:getProperty>

```
1 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
2 <html>
3 <head>
4 <meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
5 <title>로그인</title>
6 </head>
7 <body>
8     회원정보 입력 화면<p>
9
10    <form action="jspActionTag.jsp" method="post">
11
12        이름 <input type="text" name="name"><p>
13        나이 <input type="password" name="age"><p>
14
15        <input type="submit" value="전송">
16        <input type="reset" value="취소">
17
18    </form>
19
20 </body>
21 </html>
```

```
1 package model.dto;
2
3 public class People {
4     private String name;
5     private int age;
6
7     public People() {
8         super();
9     }
10    //...종료
11}
```

```
1 <%@ page language="java" contentType="text/html; charset=EUC-KR"
2     pageEncoding="EUC-KR"%>
3 <%@ page import="model.dto.People" %>
4 <% request.setCharacterEncoding("euc-kr"); %>
5
6 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
7 <html>
8 <head>
9 <meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
10 <title>JSP Action tag 학습</title>
11 </head>
12 <body>
13     <h3>JSP Action Tag 예제</h3><p><br>
14
15     1. 자바빈 객체 생성 tag - jsp:useBean<br><br>
16     <jsp:useBean id="people" class="model.dto.People" />
17
18     2. 매개변수값 자동 대입 tag - jsp:setProperty<br><br>
19     <jsp:setProperty property="name" name="people"/>
20     <jsp:setProperty property="age" name="people"/>
21
22     People의 setName(request.getParameter("name"))
23
24     3. 자바빈 객체 변수값 출력 tag - jsp:getProperty<br><br>
25     이름 : <jsp:getProperty property="name" name="people"/><br>
26     나이 : <jsp:getProperty property="age" name="people"/>
27
28     <br><br>
29 </body>
30 </html>
```

JSP Action Tag 예제

회원정보 입력 화면

이름

나이

JSP Action Tag 예제

1. 자바빈 객체 생성 tag - `jsp:useBean`
2. 매개변수값 자동 대입 tag - `jsp:setProperty`
3. 자바빈 객체 변수값 출력 tag - `jsp:getProperty`
이름 : 김혜경
나이 : 40

Expression Language[EL tag]

1. 필요성

- jsp에서 자바 코드를 최소화 하자는 취지에서 제공

- view 위주로 개발 권장

2. 기능

- 데이터를 브라우저에 출력

3. **`${출력데이터}`**

Expression Language

tag	특징
	화면에 출력하는 기능을 제공 Scripting tag 감소효과 JSP2.0 스펙에 추가 빈즈,맵,리스트,배열 유형의 객체의 속성이나 원소를 접근하여 화면에 출력 스크립팅 코드를 대신할 수 있음
<code>\${}</code>	JSTL과 함께 쓸 경우 많은 스크립팅 코드를 대신 할 수 있어서 유용함
	브래킷(<code>[]</code>) 연산자 왼편에는 map, beans, array, list 변수 가 올 수 있음 브래킷(<code>[]</code>) 연산자 안에는 key값(map), 속성명(beans),index(array, list)가 올 수 있음 도트(<code>.</code>) 연산자 왼편에는 map, beans 변수 가 올 수 있음 도트(<code>.</code>) 연산자 안에는 key값(map), 속성명(beans)이 올 수 있음

Expression Language의 내장객체(1/3)

내장 객체	역할
pageScope	pageScope에 저장된 속성 Map
requestScope	requestScope에 저장된 속성 Map
sessionScope	sessionScope에 저장된 속성 Map
applicationScope	applicationScope에 저장된 속성 Map
param	request parameter 속성 Map
paramValues	request parameter(multi value) 속성 Map
header	request header 속성 Map
headerValues	request header(multi value) 속성 Map
cookie	cookie 속성 Map
initParam	context init parameter 속성 Map
pageContext	pageContext Beans

The diagram shows a large curly bracket on the right side of the table, grouping the first ten rows (pageScope through initParam) and pointing to a box labeled "Map". A horizontal line connects the "pageContext" row to a box labeled "Beans".

Expression Language Syntax

EL 내장 객체	속성	syntax
pageScope	page scope에 바인딩된 속성	<code>\${pageScope...}</code>
requestScope	request scope에 바인딩된 속성	<code>\${requestScope...}</code>
sessionScope	session scope에 바인딩된 속성	<code>\${sessionScope...}</code>
applicationScope	application scope에 바인딩된 속성	<code>\${applicationScope...}</code>
param	<code>getParameter()</code> 와 동일	<code>\${param...}</code>
paramValues	<code>getParameterValues()</code> 와 동일	<code>\${paramValues...}</code>
cookie	Cookie 객체	<code>\${cookie...}</code>
initParam	<code>getInitParameter()</code> 와 동일	<code>\${initParam...}</code>
...		

Expression Language syntax

```
40 <!-- 중략 -->
41 <table border="1">
42   <tr>
43     <th><b>EL Expression</b></th>
44     <th><b>Result</b></th>
45   </tr>
46   <tr>
47     <td>\${param.foo}</td>
48     <td>\${param.foo}</td>
49   </tr>
50   <tr>
51     <td>\${paramValues.hobby[0]}</td>
52     <td>\${paramValues.hobby[0]}</td>
53   </tr>
54   <tr>
55     <td>\${paramValues["hobby"][1]}</td>
56     <td>\${paramValues["hobby"][1]}</td>
57   </tr>
58   <tr>
59     <td>\${header["accept"]}</td>
60     <td>\${header["accept"]}</td>
61   </tr>
62   <tr>
63     <td>\${initParam["adminEmail"]}</td>
64     <td>\${initParam["adminEmail"]}</td>
65   </tr>
66   <tr>
67     <td>\${cookie.userId.value}</td>
68     <td>\${cookie.userId.value}</td>
69   </tr>
70 </table>
71 <!-- 중략 -->
```

JSP 2.0 Expression Language – Implicit Objects

EL Expression	Result
<code>\\${param.foo}</code>	happy
<code>\\${paramValues.hobby[0]}</code>	movie
<code>\\${paramValues["hobby"][1]}</code>	swimming
<code>\\${header["accept"]}</code>	image/gif, image/jpeg, image/pjpeg, application/x-ms-application, application/vnd.ms-xpsdocument, application/xhtml+xml, application/x-ms-xbap, application/x-shockwave-flash, application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword, */*
<code>\\${initParam["adminEmail"]}</code>	admin@my.com
<code>\\${cookie.userId.value}</code>	java

Expression Language 연산자 - 기본 연산자(1/3)

*** 산술 연산자 ***

더하기:	+
빼기:	-
곱하기:	*
나누기:	/와 div
나머지:	%와 mod

*** 관계 연산자 ***

등호:	==와 eq
부등호:	!=와 ne
~보다 작다:	<와 lt
~보다 크다:	>와 gt
~보다 작거나 같다:	<=와 le
~보다 크거나 같다:	>=와 ge

*** 논리 연산자 ***

AND:	&&와 and
OR:	와 or
NOT:	!와 not

Expression Language 연산자 - 기본 연산자

```
13 <!-- 중략 -->
14 <tr>
15     <th><b>EL Expression</b></th>
16     <th><b>Result</b></th>
17 </tr>
18 <tr>
19     <td>\${1}</td>
20     <td>${1}</td>
21 </tr>
22 <tr>
23     <td>\${1 + 2}</td>
24     <td>${1 + 2}</td>
25 </tr>
26 <tr>
27     <td>\${1.2 + 2.3}</td>
28     <td>${1.2 + 2.3}</td>
29 </tr>
30 <tr>
31     <td>\${1.2E4 + 1.4}</td>
32     <td>${1.2E4 + 1.4}</td>
33 </tr>
34 <tr>
35     <td>\${-4 - 2}</td>
36     <td>${-4 - 2}</td>
37 </tr>
38 <!-- 중략 -->
```

EL Expression	Result
<code>\${1}</code>	1
<code>\${1 + 2}</code>	3
<code>\${1.2 + 2.3}</code>	3.5
<code>\${1.2E4 + 1.4}</code>	12001.4
<code>\${-4 - 2}</code>	-6
<code>\${21 * 2}</code>	42
<code>\${3/4}</code>	0.75
<code>\${3 div 4}</code>	0.75
<code>\${3/0}</code>	Infinity
<code>\${10%4}</code>	2
<code>\${10 mod 4}</code>	2
<code>\${(1==2) ? 3 : 4}</code>	4

Expression Language 예제

```
1 <%@ page language="java" contentType="text/html; charset=EUC-KR"
2   pageEncoding="EUC-KR"%>
3 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.c
4 <html>
5 <head>
6 <meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
7 <title>Insert title here</title>
8 </head>
9 <body>
10  <h3>EL Tag 예제</h3><p><hr>
11    요청객체에 데이터 저장해서 다음 page에서 출력하기
12    <% request.setAttribute("message", "el Tag에서 사용될 데이터"); %>
13
14    <!-- jsp Action tag로 page이동[포워드 방식] | - %>
15    <jsp:forward page="elTag2.jsp"/>
16 </body>
```

```
1 <%@ page language="java" contentType="text/html; charset=EUC-KR"
2   pageEncoding="EUC-KR"%>
3 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www
4 <html>
5 <head>
6 <meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
7 <title>Insert title here</title>
8 </head>
9 <body>
10  <h3>EL Tag 예제</h3><p><hr>
11    EL Tag로 데이터 출력<br><br>
12    <font color="red">${requestScope.message}</font>
13 </body>
14 </html>
```

EL Tag 예제

요청객체에 데이터 추출해서 데이터 획득후 출력

EL Tag로 데이터 출력

el Tag에서 사용될 데이터

Expression Language 연산자 - 도트(.) 연산자(1/3)

```
4 <%
5   HashMap<String,String> list = new HashMap<String,String>();
6   list.put("first","Walk to remember");
7   list.put("second","인생은 아름다워");
8
9   request.setAttribute("movieList",list);
10
11   Person p = new Person("이동욱",3);
12   request.setAttribute("mySon",p);
13 %>
14 <html>
15 <!-- 중략 -->
16 <body>
17   <h1>JSP 2.0 Expression Language - 속성(Map , Bean) </h1>
18   <table border="1">
19     <tr>
20       <th><b>EL Expression</b></th>
21       <th><b>Result</b></th>
22     </tr>
23     <tr>
24       <td>\${movieList.first}</td>
25       <td>\${movieList.first}</td>
26     </tr>
27     <tr>
28       <td>\${movieList["second"]} </td>
29       <td>\${movieList["second"]} </td>
30     </tr>
31   </table>
32   <br>
33   <table border="1">
34     <tr>
35       <th><b>EL Expression</b></th>
36       <th><b>Result</b></th>
37     </tr>
38     <tr>
39       <td>\${mySon.name}</td>
40       <td>\${mySon.name}</td>
41     </tr>
42     <tr>
43       <td>\${mySon["age"]} </td>
44       <td>\${mySon["age"]} </td>
45     </tr>
46   </table>
```

JSP 2.0 Expression Language - 속성(Map , Bean)

EL Expression	Result
\\${movieList.first}	Walk to remember
\\${movieList["second"]}	인생은 아름다워

EL Expression	Result
\\${mySon.name}	이동욱
\\${mySon["age"]}	5

Expression Language 연산자 - 브래킷([]) 연산자(2/3)

```
1 <%@ page language="java" contentType="text/html; charset=EUC-KR"
2   import="java.util.*"
3   pageEncoding="EUC-KR"%>
4 <!-- 중략 -->
5 <%
6
7   ArrayList<String> list = new ArrayList<String>();
8   list.add("Walk to remember");
9   list.add("과속 스캔들");
10  list.add("인생은 아름다워");
11
12  //String[] list = {"Walk to remember1","과속 스캔들1",
13  request.setAttribute("movieList",list);
14
15 %>
16 <body>
17   <h1>JSP 2.0 Expression Language - 속성(List) </h1>
18   <table border="1">
19     <tr>
20       <th><b>EL Expression</b></th>
21       <th><b>Result</b></th>
22     </tr>
23     <tr>
24       <td>\${movieList[0]}</td>
25       <td>\${movieList[0]}</td>
26     </tr>
```

```
27   <tr>
28     <td>\${movieList[1]}</td>
29     <td>\${movieList[1]}</td>
30   </tr>
31   <tr>
32     <td>\${movieList["2"]}</td>
33     <td>\${movieList["2"]}</td>
34   </tr>
35 </table>
36 <br>
37 <table border="1">
38   <tr>
39     <th><b>EL Expression</b></th>
40     <th><b>Result</b></th>
41   </tr>
42   <tr>
43     <td>\${requestScope.movieList[0]}</td>
44     <td>\${requestScope.movieList[0]}</td>
45   </tr>
46   <tr>
47     <td>JSP 2.0 Expression Language
48     <td>
49   </tr>
50   <tr>
51     <td>\${movieList[0]}</td>
52     <td>Walk to remember
53   </tr>
54   <tr>
55     <td>\${movieList[1]}</td>
56     <td>과속 스캔들
57   </tr>
58   <tr>
59     <td>\${movieList["2"]}</td>
60     <td>인생은 아름다워
61   </tr>
62 </table>
63 </body>
64 </html>
```

EL Expression	Result
\\${movieList[0]}	Walk to remember
\\${movieList[1]}	과속 스캔들
\\${movieList["2"]}	인생은 아름다워

EL Expression	Result
\\${requestScope.movieList[0]}	Walk to remember
\\${requestScope.movieList[1]}	과속 스캔들
\\${requestScope.movieList["2"]}	인생은 아름다워

JSP Standard Tag Library

JSTL의 개요(1/2)

특징

JSP Standard Tag Library

EL 이나 표준 액션으로 처리하기 힘든 부분을 담당하기 위해 확장된 표준 Custom Tag Library

웹 애플리케이션 기능 중 일반화된 기능(반복과 조건 분기처리, 데이터 관리 포맷, XML조작, DB 액세스)을 미리 구현한 Tag Library

JSP2.0부터 JSP스펙에 포함

종 류	CORE	주로 프리젠테이션 부분에서 효과적으로 사용 데이터관리, 반복, 조건에 관한 태그 지원
	XML	Xml 파싱과 생성에 관한 지원
	Format(I18N)	Internationalization 숫자와 날짜를 포맷하는 기능
	SQL	JDBC를 이용한 DB 처리 기능

JSTL 태그의 종류

분류	URI	prefix	적용 예
core	http://java.sun.com/jsp/jstl/core	c	<c:tagname .. >
xml	http://java.sun.com/jsp/jstl/xml	x	<x:tagname .. >
i18n	http://java.sun.com/jsp/jstl/fmt	fmt	<fmt:tagname .. >
sql	http://java.sun.com/jsp/jstl/sql	sql	<sql:tagname .. >

JSTL의 개요(2/2)

1. jstl.jar, standard.jar 라이브러리 필요
 1. %APPLICATION_ROOT%\WEB-INF\lib 에 복사
2. <http://tomcat.apache.org/taglibs/standard/>

Apache Taglibs - Apache Standard Taglib: JSP(tm) Standard Tag Library (JSTL) implementations - Windows Internet Explorer

http://tomcat.apache.org/taglibs/standard/

Google

스마트홈 Bdtm 검색 알패스 즐겨찾기 옥션 문자 캡쳐

즐거찾기 Apache Taglibs - Apache Standard Tagli...

ApacheCon NORTH AMERICA 2011
7-11 November
Vancouver, BC

Apache Taglibs
Home
Using
Taglib Tutorial
News Archives
Wiki

Digging into Apache Taglibs
Mailing Lists
Bugs
Building
CI

The Taglibs

Taglibs

Standard Taglib

JSP(tm) Standard Tag Library implementations

Apache hosts the Apache Standard Taglib, an implementation of the [JSP Standard Tag Library \(JSTL\)](#) specification. Various versions are available.

Version	JSTL version	Requirements	Getting the Taglib
Standard 1.2	JSTL 1.2 (not yet JCP approved)	Servlet 2.5, JavaServer Pages 2.1	download
Standard 1.1	JSTL 1.1	Servlet 2.4, JavaServer Pages 2.0	download
Standard 1.0	JSTL 1.0	Servlet 2.3, JavaServer Pages 1.2	download

News

인터넷 | 보호 모드: 설정

100%

JSTL 태그의 종류(3/3)

코어(Core) 라이브러리 일반적인 것

```
<c:out>
<c:set>
<c:remove>
<c:catch>
```

조건

```
<c:if>
<c:choose>
<c:when>
<c:otherwise>
```

URL 관련

```
<c:import>
<c:url>
<c:redirect>
<c:param>
```

반복

```
<c:forEach>
<c:forEachToken>
```

포매팅 라이브러리 국제화

```
<fmt:message>
<fmt:setLocale>
<fmt:bundle>
<fmt:setBundle>
<fmt:param>
<fmt:requestEncoding>
```

포매팅

```
<fmt:timeZone>
<fmt:setTimeZone>
<fmt:formatNumber>
<fmt:parseNumber>
<fmt:parseDate>
```

SQL 라이브러리

데이터베이스 접근

```
<sql:query>
<sql:update>
<sql:setDataSource>
<sql:param>
<sql:dateParam>
```

XML 라이브러리 코어 XML 액션

```
<x:parse>
<x:out>
<x:set>
```

XML 흐름 제어

```
<x:if>
<x:choose>
<x:when>
<x:otherwise>
<x:forEach>
```

변환 액션

```
<x:transform>
<x:param>
```

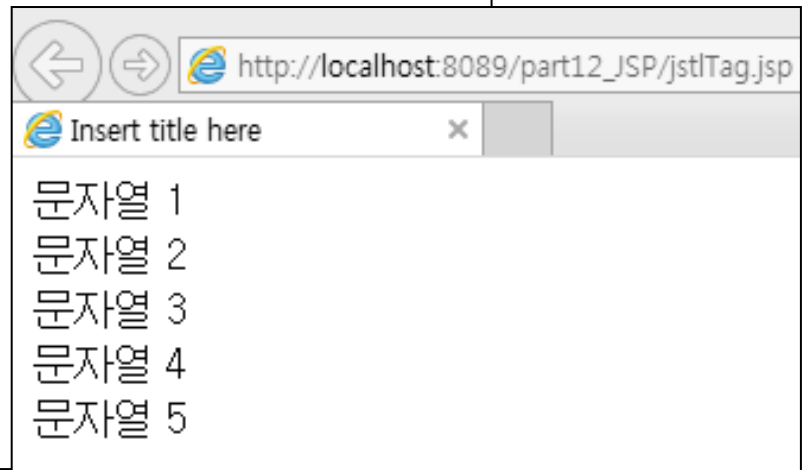
출처:Head First Servlets & JSP

JSTL 주요 핵심 tag

종 류	특 징
<c:forEach>	반복 <code><c:forEach begin="1" end="10" varStatus="values"></code>
<c:set>	데이터 특정 scope에 저장 setAttribute와 동일
<c:remove>	특정 scope에 저장된 데이터 삭제
<c:if>	조건식
<c:choose> <c:when> <c:otherwise>	다중 조건식
...	

JSTL 예제

```
1 <%@ page language="java" contentType="text/html; charset=EUC-KR"
2   pageEncoding="EUC-KR"%>
3 <%@ page import="java.util.ArrayList" %>
4 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
5
6 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org
7 <html>
8 <head>
9 <meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
10 <title>Insert title here</title>
11 </head>
12 <body>
13   <%
14     ArrayList<String> dataAll = new ArrayList<String>();
15     dataAll.add("문자열 1");
16     dataAll.add("문자열 2");
17     dataAll.add("문자열 3");
18     dataAll.add("문자열 4");
19     dataAll.add("문자열 5");
20     pageContext.setAttribute("all", dataAll);
21   %>
22   <c:forEach items="${all}" var="jstlTagVar">
23     ${jstlTagVar}<br>
24   </c:forEach>
25 </body>
26 </html>
```



Core 태그 - <c:set>, <c:remove> (1/4)

1. <c:set>

1. setAttribute()와 동일
2. scope = page | request | session | application

```
<c:set var="변수명" value="변수값"  
      scope="page | request | session | application" />
```

```
<c:set value="변수값" target="자바빈객체명 또는 Map객체명"  
      property="프로퍼티명" />
```

2. <c:remove>

1. removeAttribute()와 같은 역할
2. scope = page | request | session | application

```
<c:remove var="변수명"  
          scope="page | request | session | application" />
```

Core 태그 - <c:set>, <c:remove>

```
4 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
5 jstl set tag를 활용한 데이터값 저장<br>
6
7 <c:set scope="request" value="이진규" var="name"/>
8 ${requestScope.name}<br>
9
10 <c:remove var="name"/>
11 remove tag 실행후 데이터값 : ${requestScope.name}<br>
```

실행 결과

jstl set tag를 활용한 데이터값 저장
이진규
remove tag 실행후 데이터값 :

Core 태그 - <c:if> (1/3)

I. <c:if> 태그

I. 조건문 역할

```
<c:if test="조건식" var="변수명"  
scope="page | request | session | application" />
```

Core 태그 - <c:if> (2/3)

```
12 jstl의 조건식 tag[EL 활용] <br>
13 <c:if test="${a} == 'b' }">
14     조건이 true인 경우만 출력
15 </c:if>
16 <br><hr><br>
17 <%
18     session.setAttribute("v", null);
19     out.println(session.getAttribute("v"));
20 %>
21 <Br>
22 -${sessionScope.v}-<br>
23 <!-- if jstl tag로 session의 v 값이 null인 경우 "안녕하세요" 출력 -->
24 <c:if test="${sessionScope.v} == null">
25     안녕하세요 1 <br>
26 </c:if>
27 <!-- null 아닌 경우에만 출력되는 문법 -->
28 <c:if test="${sessionScope.v} != null">
29     안녕하세요 2 <br>
30 </c:if>
31 <!-- null 즉 blank인 경우에만 출력되는 문법 -->
32 <c:if test="${empty sessionScope.v}">
33     안녕하세요 3 <br>
34 </c:if>
35 <!-- null 즉 blank가 아닌 경우에만 출력되는 문법 -->
36 <c:if test="${not empty sessionScope.v}">
37     안녕하세요 4 <br>
38 </c:if>
```

jstl의 조건식 tag[EL 활용]

null

--

안녕하세요 1

안녕하세요 3

표현언어의 개요 - null처리

1. EL은 null값에 대한 특별한 처리를 해줌으로써 null에 대한 체크가 필요 없음
2. null값을 연산에 따라 0, false, ""(빈 문자열) 로 처리해 줌

```
6 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
7 <%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions" %>
8
9 ArrayList객체는 존재 단, 저장된 요소 객체들은 없음, 이 경우 조건 체크를 어떻게?<br>
10 <%
11     ArrayList noData = new ArrayList();
12     request.setAttribute("noData", noData);
13 %>
14 1. ArrayList내의 데이터 개수 - ${fn:length(noData)}<br>
15
16 <c:if test="${fn:length(noData) == 0 }">
17     2. length() 함수를 활용한 데이터 개수 확인 - 데이터 없다<br>
18 </c:if>
19
```

```
20 <c:if test="${not empty noData}">
21     3. not empty 연산자를 활용한 데이터 개수 확인 - 데이터 있음
22 </c:if>
23
24 <c:if test="${empty noData}">
25     4. empty 연산자를 활용한 데이터 개수 확인 - 데이터 없다
26 </c:if>
```

http://localhost/step12_jstl/jspStep03jstlForEach.jsp

ArrayList객체는 존재 단, 저장된 요소 객체들은 없음, 이 경우 조건 체크를 어떻게?

1. ArrayList내의 데이터 개수 - 0

2. length() 함수를 활용한 데이터 개수 확인 - 데이터 없다

3. not empty 연산자를 활용한 데이터 개수 확인 - 데이터 있음

4. empty 연산자를 활용한 데이터 개수 확인 - 데이터 없다

Core 태그 - <c:forEach> (1/5)

I. <c:forEach> 태그

I. 반복문 역할

```
<c:forEach      items="반복대상객체명"  
                begin="시작값" end="종료값" step="증가값"  
                var="변수명" varStatus="반복상태값의변수명" />
```

Core 태그 - <c:forEach> (4/5)

<pre>18 jstl의 반복 tag인 forEach (자바의 반복문과 동일)

 19 20 1. 1~10까지 출력
 21 <c:forEach begin="1" end="10" step="2" 22 var="naver" varStatus="v"> 23 \${v.index} - \${naver} - \${v.count }
 24 </c:forEach> 25 26
<hr>
 27 28 2. 1~10까지의 합 결과값만 출력
 29 set과 forEach
 30 <c:forEach begin="1" end="10" varStatus="values"> 31 <c:set var="sum" value="\${sum + values.count}"/> 32 </c:forEach> 33 \${sum} 34</pre>	<p>jstl의 반복 tag인 forEach (자바의 반복문과 동일)</p> <p>1. 1~10까지 출력</p> <p>1 - 1 - 1 3 - 3 - 2 5 - 5 - 3 7 - 7 - 4 9 - 9 - 5</p> <hr/> <p>2. 1~10까지의 합 결과값만 출력 set과 forEach 55</p>
--	---

Core 태그 - <c:forEach>(

```
37 3. 자바 객체를 forEach tag로 활용하기<br>
38<%
39     People p1 = new People("master", 10);
40     People p2 = new People("tester", 20);
41     People p3 = new People("vip", 15);
42     ArrayList all = new ArrayList();
43     all.add(p1);
44     all.add(p2);
45     all.add(p3);
46     session.setAttribute("all", all);
47 %>
48<c:forEach items="${sessionScope.all}" var="data">
49     ***${data.id}***<br>
50 </c:forEach>
51
52 <br><hr><br>
53<%
54     HashMap map = new HashMap();
55     map.put("p1", p1);
56     map.put("p2", p2);
57     map.put("p3", p3);
58     application.setAttribute("m", map);
59 %>
60 HashMap의 데이터 출력<br>
61<c:forEach items="${applicationScope.m}" var="p">
62     ${p.key}-${p.value}-${p.value.id}<br>
63 </c:forEach>
```

3. 자바 객체를 forEach tag로 활용하기

master
tester
vip

HashMap의 데이터 출력

p3-People [id=vip, age=15]-vip
p2-People [id=tester, age=20]-tester
p1-People [id=master, age=10]-master

Formatting 태그(1/3)

1. <fmt:formatNumber>태그

- 1. 숫자형식을 표현할 때 사용

```
<fmt:formatNumber value="수치값"  
  type="number | percent | currency" currencySymbol="통화기호"  
  pattern="형식" ... />
```

2. <fmt:formatDate>태그

- 1. 날짜형식을 표현할 때 사용

```
<fmt:formatDate value="Date와Time"  
  type="time | date | both"  
  pattern="형식" ... />
```

Formatting 태그(2/3)

```
1 <%@ page language="java" contentType="text/html; charset=EUC-KR"
2   pageEncoding="EUC-KR" import="java.util.Date" %>
3 <%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
4 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
5 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
6   "http://www.w3.org/TR/html4/loose.dtd">
7 <html>
8 <head>
9 <meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
10 <title>JSTL Core - fmt</title>
11 </head>
12 <body>
13   <H2>숫자형식 지정</H2>
14   숫자 : <fmt:formatNumber value="123456.78" type="number"/> <br/>
15   통화 : <fmt:formatNumber value="7700000" type="currency" currencySymbol="$"/> <br/>
16   패턴 : <fmt:formatNumber value="123456.789" pattern=".00"/> <br/>
17
18   <c:set var="currentDate" value="<%=new Date() %>"/>
19   <H2>날짜형식 지정</H2>
20   날짜 : <fmt:formatDate value="${currentDate}" type="date"/> <br/>
21   시간 : <fmt:formatDate value="${currentDate}" type="time"/> <br/>
22   날짜+시간 : <fmt:formatDate value="${currentDate}" type="both" /> <br/>
23 </body>
24 </html>
```

숫자형식 지정

숫자 : 123,456.78
통화 : \$ 7,700,000.00
패턴 : 123456.79

날짜형식 지정

날짜 : 2011. 5. 9
시간 : 오후 1:07:07
날짜+시간 : 2011. 5. 9 오후 1:07:07



Java Script & jQuery

학습내용

Java Script 개요 및 문법

DOM(Document Object Model)

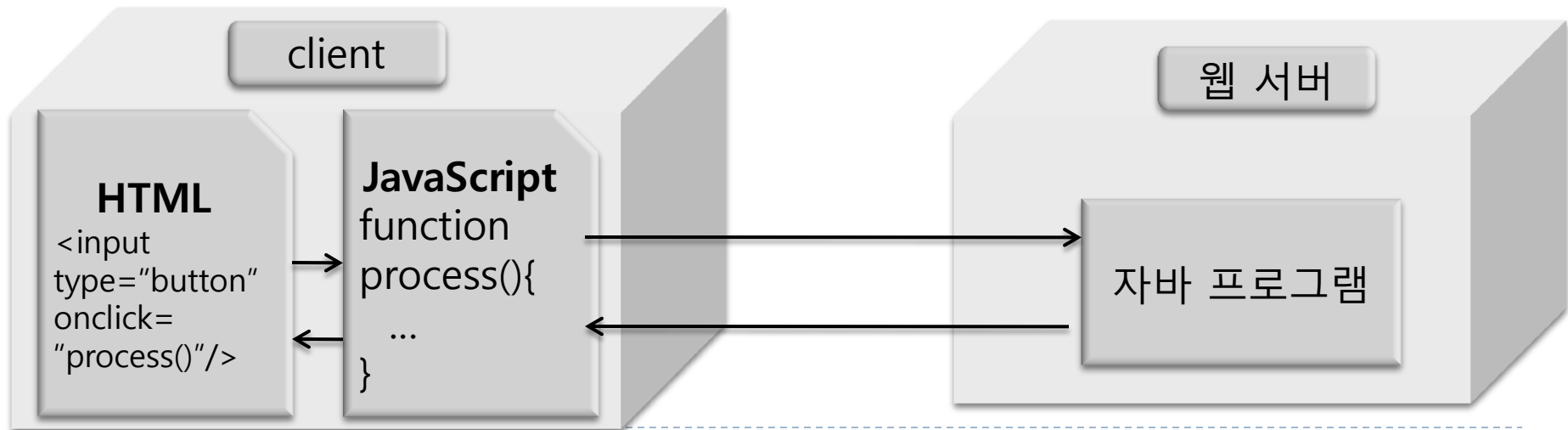
jQuery

jQuery Ajax Programming

Java Script 개요 및 문법

JavaScript 개요

1. Ajax에서의 자바 스크립트
 1. 사용자의 요청과 로직 사이에서 이들을 연계 시켜주는 기능
 2. 입출력에 관한 동작을 처리
 3. 실행 process
 1. 사용자들의 요구인 이벤트를 처리하는 이벤트 핸들러 호출
 2. 핸들러는 사용자의 요구를 받고, 서버에 요청
 3. 서버단 자바 프로그램이 요청 처리 후 응답
 4. 응답한 결과를 받아서 화면에 HTML 태그로 표시



JavaScript 문법 - HTML 문서에 적용 문법

I. 외부 자바스크립트 파일을 HTML에 적용시

```
<script language="스크립트 언어이름" src="외부 스크립트 파일의 URL">  
    var varName;  
    function 함수명(){ ... }  
    주석[ // or /* */ ]  
</script>
```

I. HTML 내부에서의 자바스크립트

```
<script type="text/javascript">  
    var varName;  
    function 함수명(){...}  
    주석[ // or /* */ ]  
</script>
```


JavaScript 문법 – 주요 데이터 타입

데이터 타입		설명	
기본 데이터 타입	숫자	정수, 실수	
	문자열	문자열 타입, " " 또는 ' ' 표현	
	boolean	참(true)과 거짓(false)	
	null	아무런 값도 할당하지 않은 알 수 없다는 의미	
레퍼런스 타입	내장 객체	window	웹 브라우저 창 및 프레임을 표시하기 위한 최상위 객체
		document	HTML 파일 및 기술된 문서의 정보 제공 및 그에 대한 작업을 수행하는 객체로 Layer, Image, Form등의 하위 객체 보유
		Date	날짜와 시간의 정보를 얻어 낼 때 사용하는 객체
		Array	배열을 생성할 때 사용하는 객체
		RegExp	정규 표현식 객체로 정규 표현식의 패턴을 보유
		...	http://www.w2school.com 참조

JavaScript 문법 – 변수 문법과 특징

1. 문법

1. 변수 타입 선언 없이 변수명만 기술

```
var 변수명;  
var 변수명 = 값;
```

2. 특징

1. 값 대입 시점에 변수 타입 자동 설정

JavaScript 문법 - 선언 위치에 따른 변수 종류

1. 전역 변수

1. 함수 외부에서 선언
2. 함수 내에서 선언시 var 키워드 없이 선언
3. 프로그램 전역에서 사용 가능

2. 지역 변수

1. 함수내에서 var 키워드 사용하여 선언
2. 선언된 함수 내에서만 사용

```
4 <script>
5   var myVar="전역변수";
6
7   function myFunction(){
8       var myVar="지역변수";
9       global = "나도 전역변수";
0       document.write("myVar : " + myVar + "<br>"); // myVar은 지역변수
1       document.write("this.myVar : " + this.myVar + "<br>"); // this.myVar은 전역변수
2   }
3 </script>
```

Java script 문법 – 함수 종류

1. 사용자 정의 함수

1. 발생한 이벤트를 처리하기 위해 사용자가 정의한 함수

2. 내장 함수

1. 자체적으로 제공하는 특정 작업을 수행하는 내장 함수

Java script 문법 - 사용자 정의 함수

1. 문법

[구현 문법]

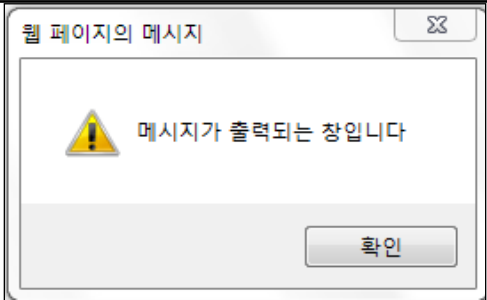
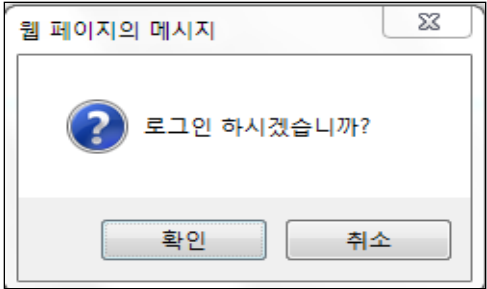
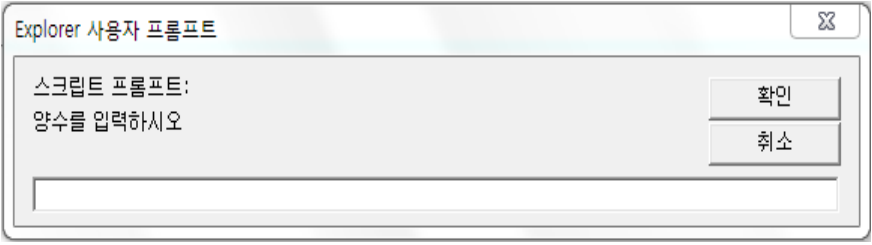
```
function 함수명( [args1, args2,..., argsn] ){  
    statement  
    [return expression;]  
}
```

[호출 문법]

1. 매개 변수가 없는 함수 호출 - 함수명();
2. 매개 변수가 있는 함수 호출 - 함수명("데이터");

1. function은 함수 선언시 필수 키워드
2. 함수명 임의 설정 가능
3. 매개 변수 선언 가능
4. 해당 함수를 호출한 코드에 "return expression" 로 값 반환

Java script 문법 – 주요 내장 함수(1/2)

함수명	설명	
alert()	메시지 출력 다이얼로그 창	
confirm()	특정 문자열을 표시하고 [확인],[취소] 버튼 제공 확인 클릭 – true 반환 취소 클릭 – false 반환	
prompt()	입력 상자를 화면에 표시해서 특정 문자열 또는 숫자값을 입력받음	

Java script 문법 – 주요 내장 함수(2/2)

함수명	설명
eval()	특정 문자열을 객체로 변환 예 : "document.myForm" 이라는 문자열을 eval("document.myForm")으로 할 경우 document.myForm 객체로 자동 변환 따라서 myForm의 하위 객체에 access 가능
escape()	문자를 인코딩할 때 사용되는 함수 영문자와 숫자는 그대로 표시
encode URIComponent()	문자열을 utf-8형식으로 변환

이벤트 - 이벤트와 이벤트 핸들러

1. 이벤트란?

1. 어플리케이션 상에서 발생하는 것으로, 어떤 작업을 수행하기 위한 동작
2. 예 : 사용자가 키 입력, 마우스 클릭...
3. 70여 가지의 이벤트 속성 제시됨

2. 이벤트 핸들러

1. 이벤트가 발생되었을 때 상응하는 작용을 감지하고 처리하는 로직
2. 메소드로 구현됨

▶ Ajax & 이벤트

1. 이벤트 기반에서 동작
2. 발생한 이벤트를 통해서 서버측에 비동기 요청 후 응답 데이터 출력

이벤트 - 주요 이벤트 속성

이벤트	설명
onBlur	입력 폼 등에서 특정 필드의 포커스가 다른 곳으로 이동했을 때
onChange	입력 폼 등에서 특정 필드의 내용이 변경 되었을 때
onClick	입력 폼 등에서 특정 필드나 단추를 클릭했을 때
onFocus	입력 폼 등에서 특정 필드의 문자 데이터가 선택되었을 때
onSubmit	입력 폼 등에서 "submit" 버튼을 클릭 했을 때
onSelect	입력 폼 등에서 특정 필드의 문자 데이터가 선택되었을 때
onLoad	웹 브라우저상에서 페이지가 읽혀졌을 때
...70여개	...

참고 사이트 - http://www.w3schools.com/jsref/dom_obj_event.asp

이벤트 – 웹 브라우저 객체와 주요 이벤트(1/2)

1. window 객체
 1. 웹 브라우저의 내장 객체들 중 최상위 객체
 2. 모든 작업들이 이 객체를 통해서 이뤄짐
2. window 객체의 이벤트들

이벤트	설명
onBlur	입력 폼 등에서 특정 필드의 포커스가 다른 곳으로 이동했을 때
onMove	윈도우를 이동했을 때
onFocus	입력 폼 등에서 특정 필드의 문자 데이터가 선택되었을 때
onLoad	웹 브라우저상에서 페이지가 읽혀졌을 때
onUnload	브라우저로 로딩된 문서가 모두 제거될 때

이벤트 – 웹 브라우저 객체와 주요 이벤트(2/2)

1. document 객체

1. HTML의 `<body>...</body>` 사이에 기술되는 내용으로 웹 브라우저에 내용이 출력되는 부분 의미
2. `window.document` 또는 `document` 로 사용

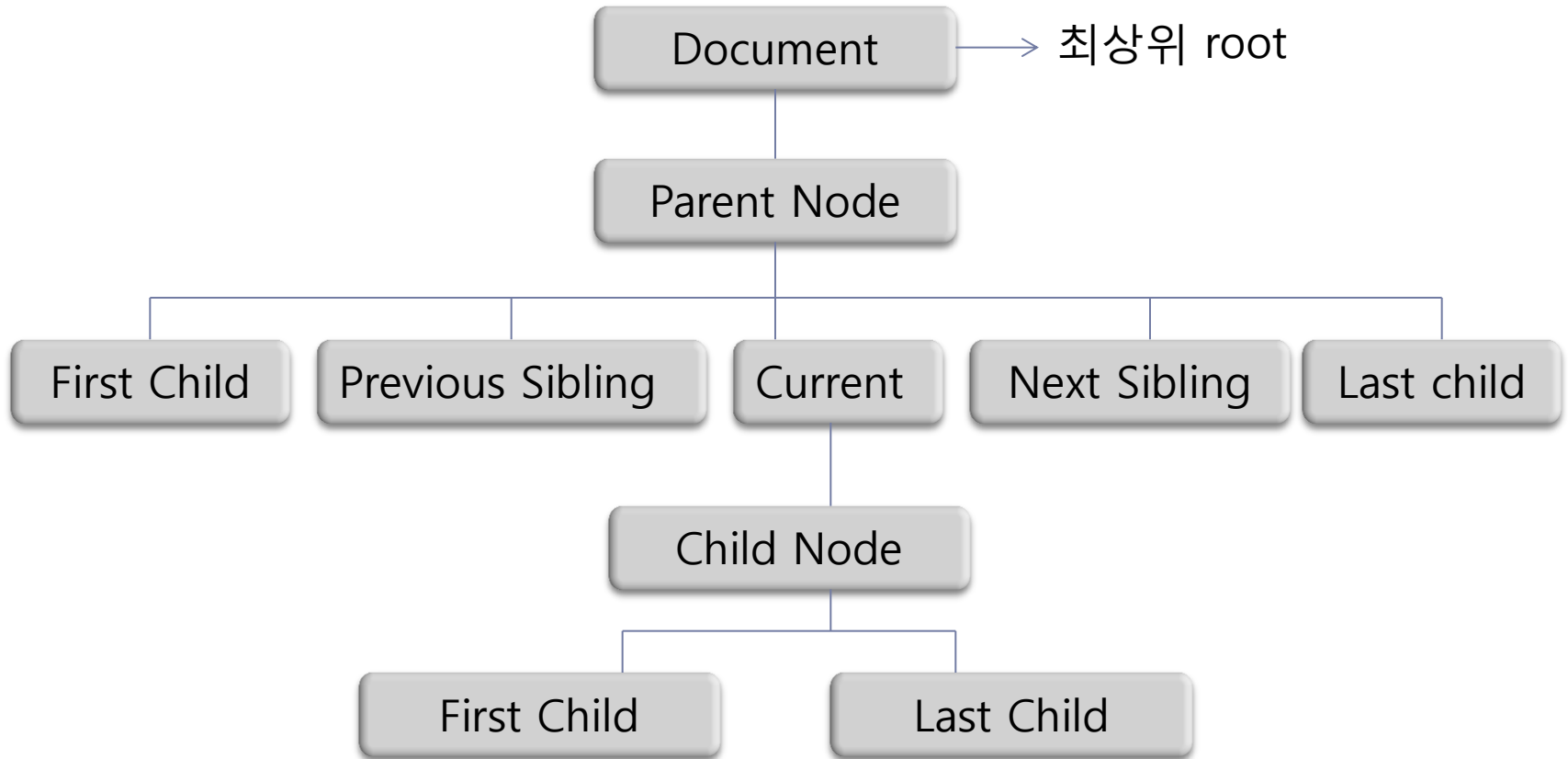
이벤트	설명
onClick	입력 폼 등에서 특정 필드나 단추를 클릭했을 때 발생
onDbClick	입력 폼 등에서 특정 필드나 단추를 더블 클릭 했을 때 발생
onKeyDown	키보드의 키를 누르는 순간 발생
onKeyUp	키보드의 키가 눌러져 있다가 키를 놓는 순간 발생
onMouseDown	마우스의 단추를 누르는 순간 발생

DOM(Document Object Model)

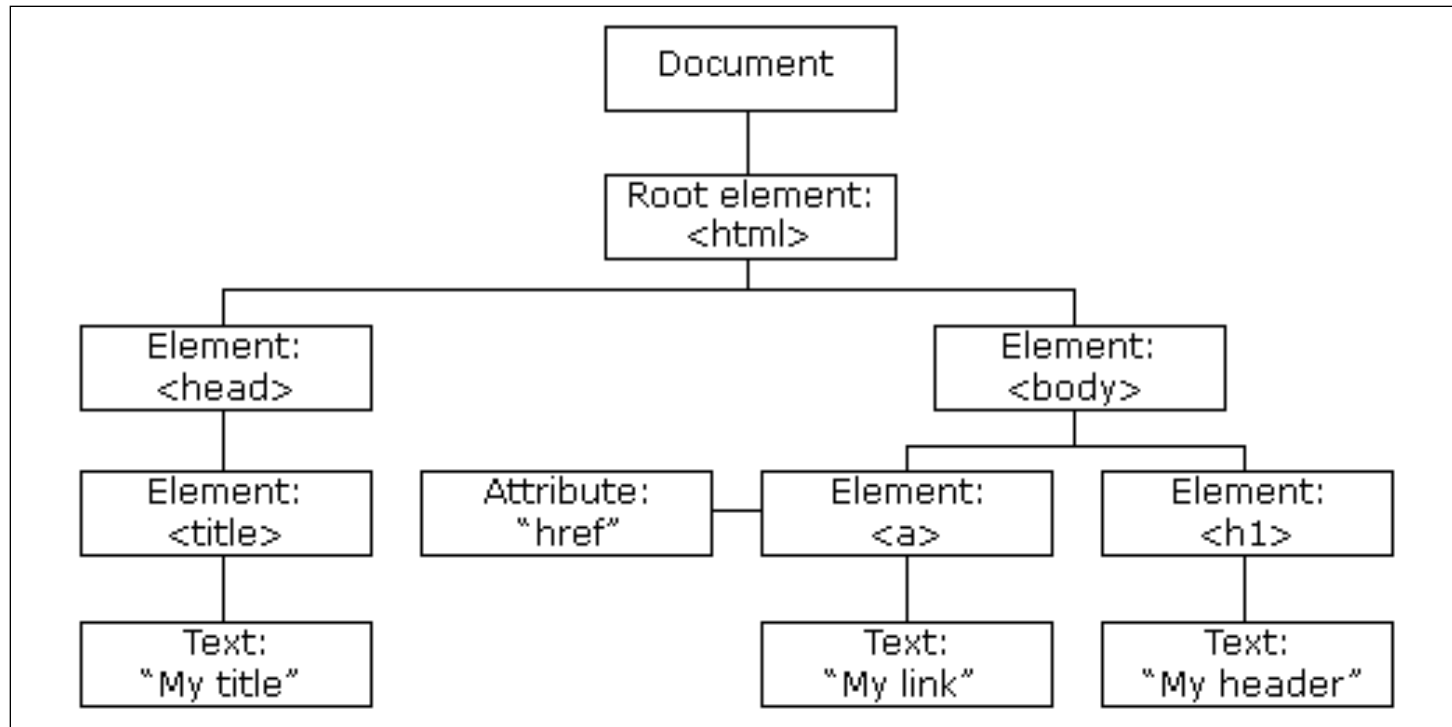
DOM[Document Object Model] 이란?

1. HTML, XML 문서를 표현하고 조작하는 표준적인 방법
2. 문서를 객체 모델로 구현해서 문서의 구조와 내용에 access하기 위한 스펙
 1. Random Access Mechanism
 2. 동적으로 문서의 수정, 삭제, 삽입, 조회 가능
 3. 메소드와 속성, 객체간의 관계를 정의하기도 함
3. W3C(World Wide Web Consortium)에서 발표
4. 플랫폼과 언어에 독립적
 1. 다양한 환경과 어플리케이션에서 사용 가능한 표준 인터페이스 제공

DOM의 기본 구조



DOM의 기본 구조 - HTML DOM Tree



발췌 사이트 - <http://www.w3schools.com/html/dom/default.asp>

DOM의 기본 구조 - DOM tree 구조 이해하기

1. Element와 Text를 tree구조로 관리
2. newline과 blank도 하나의 Text 로 인식
3. access 방식
 1. 부모 -> 자식 -> 자손 순서로 access
 2. 형제 관계에서 형, 동생 순서로 access
 3. 특정 Element 를 access 할 수도 있음

JSON의 개요

1. JavaScript Object Notation
2. JSON 은 텍스트 기반의 경량(lightweight) 데이터 변환 포맷
3. 사용자가 읽고, 쓰기 쉬워 데이터 처리가 쉬움
4. JavaScript를 기반으로 만들어졌으나, 프로그래밍 언어에 독립적인 text 형식
5. 이기종간의 데이터 교환에 적합

JSON 데이터 형식(1/4)

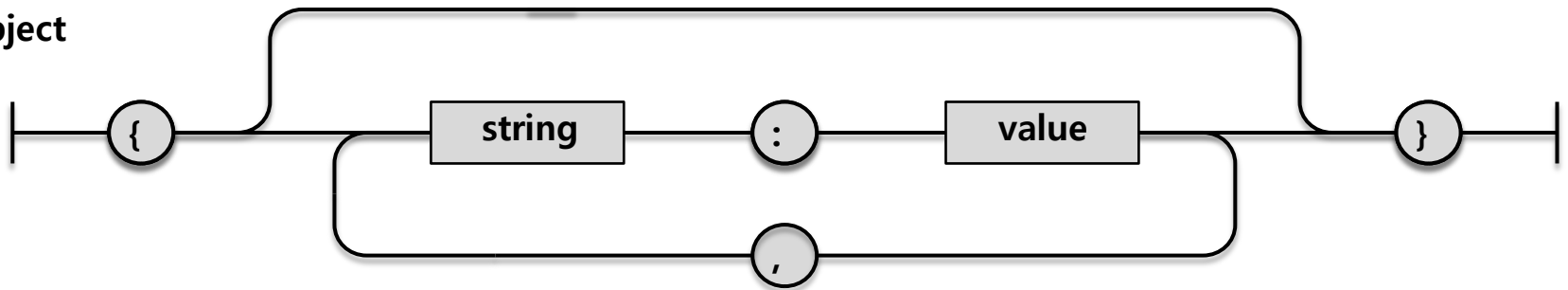
1. 컬렉션데이터 구조

1. 이름/값 쌍(pair)으로 이루어진 데이터 집합
2. 중괄호 내에 key와 value 로 구성
3. key와 value 사이에는 : 표기로 구분
4. key는 문자열 타입, value는 JSON 데이터
5. 다수의 데이터 존재할 경우 key와 value 값들은 , 표기로 구분

JSON 데이터 형식(2/4)

▶ 컬렉션데이터 구조

object



```
{  
  "name1" : value1,  
  "name2" : value2,  
  ...  
}
```

문법

```
{  
  "id" : id,  
  "pw" : pw  
}
```

예 1

```
{  
  "conversation":{  
    "decimal": "103",  
    "hyper": "&0x67"  
  }  
}
```

예 2

JSON 데이터 형식(3/4)

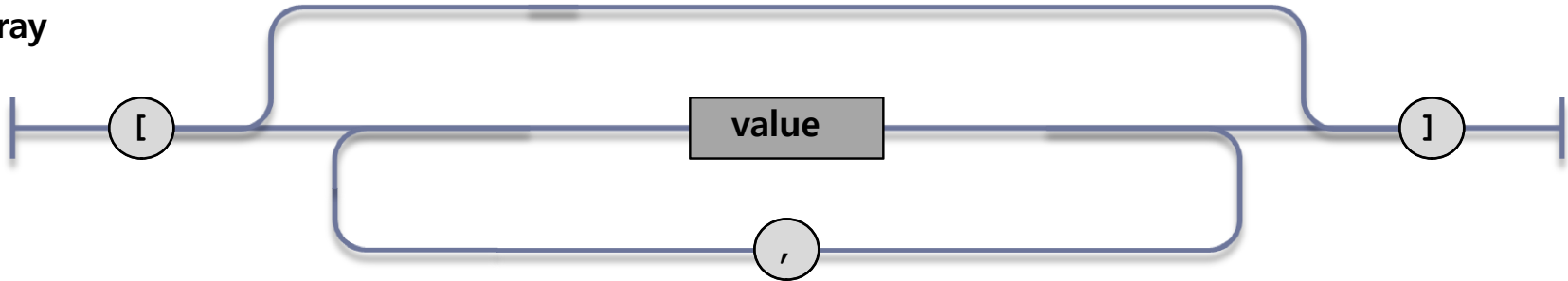
1. 배열 구조

1. 순서가 있는 데이터들의 목록
2. [(left bracket) 으로 시작해서](right bracket)으로 표현
3. 각 데이터들은 , 표기로 구분

JSON 데이터 형식(4/4)

1. 배열 구조

array



문법

```
[value1, value2, ...]
```

예 1

```
var arrayName=[id, pw]
```

예 2

```
[  
    [0, -1, 0],  
    [1, 0, 0],  
    [0, 0, 1]  
]
```

JSON 데이터 사용(1/2)

1. eval() 함수

1. JSON 형식의 문자열 데이터를 자바스크립트 객체로 변환
2. 이 함수 생략시 접근자(.) 및 []로 특정 데이터 access 불가
3. 문법

```
eval( "(" + JSON형식의 데이터 + ")" );
```

JSON 데이터 사용(2/2)

1. [] 브래킷 연산자 와 . 접근자를 활용하여 JSON 데이터 access

1. [] 연산자

1. 예 : 변수['key']

2. . 접근자

1. 예 : 변수.key

```
var jsonData = eval( "({ ko:'대한민국', fr:'프랑스', en:'미국', koValue:100, frValue:50 })" );  
  
var firstCountry = jsonData['ko']; //대한민국  
var frValue = jsonData.frValue; //50
```

jQuery

jQuery란?

1. web page를 refresh 없이 내용 변경 가능하도록 특화된 자바스크립트 라이브러리
2. 다양한 함수 제공
3. 장점
 1. DOM을 단순화, DOM 쉽게 조작 가능
 2. 개발이 쉬움
 3. jQuery library의 업그레이드 급속도로 발전
 4. html문서의 탐색, 이벤트 제어, 동적 표현 및 Ajax를 쉽고 빠르게 개발할 수 있는 기능 제공

Web Page 구성

HTML

1. Web Page의 구조 담당

JSP

Java Script

CSS

1. HTML 요소의 스타일,
즉 위치와 표현 담당

jQuery[toolkit]

1. 빠르고 간결한 개발이 가능한 Java
Script library

jQuery 개발 환경 구축

1. <http://www.jquery.org/> 사이트에서의 다운로드

Downloading jQuery

The jQuery 1.9 line has major changes from previous versions. We *strongly* recommend that you also use the jQuery Migrate plugin if you are upgrading from older versions of jQuery or need to use plugins that haven't yet been updated. Read the [jQuery 1.9 Upgrade Guide](#) and the [jQuery 1.9 release blog post](#) for more information.

[Download the compressed, production jQuery 1.9.1](#)

[Download the uncompressed, development jQuery 1.9.1](#)

[jQuery 1.9.1 release notes](#)

다운로드

2. CDN(Content Delivery Network)에 올려져 있는 jQuery 활용

1. `<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.min.js">`

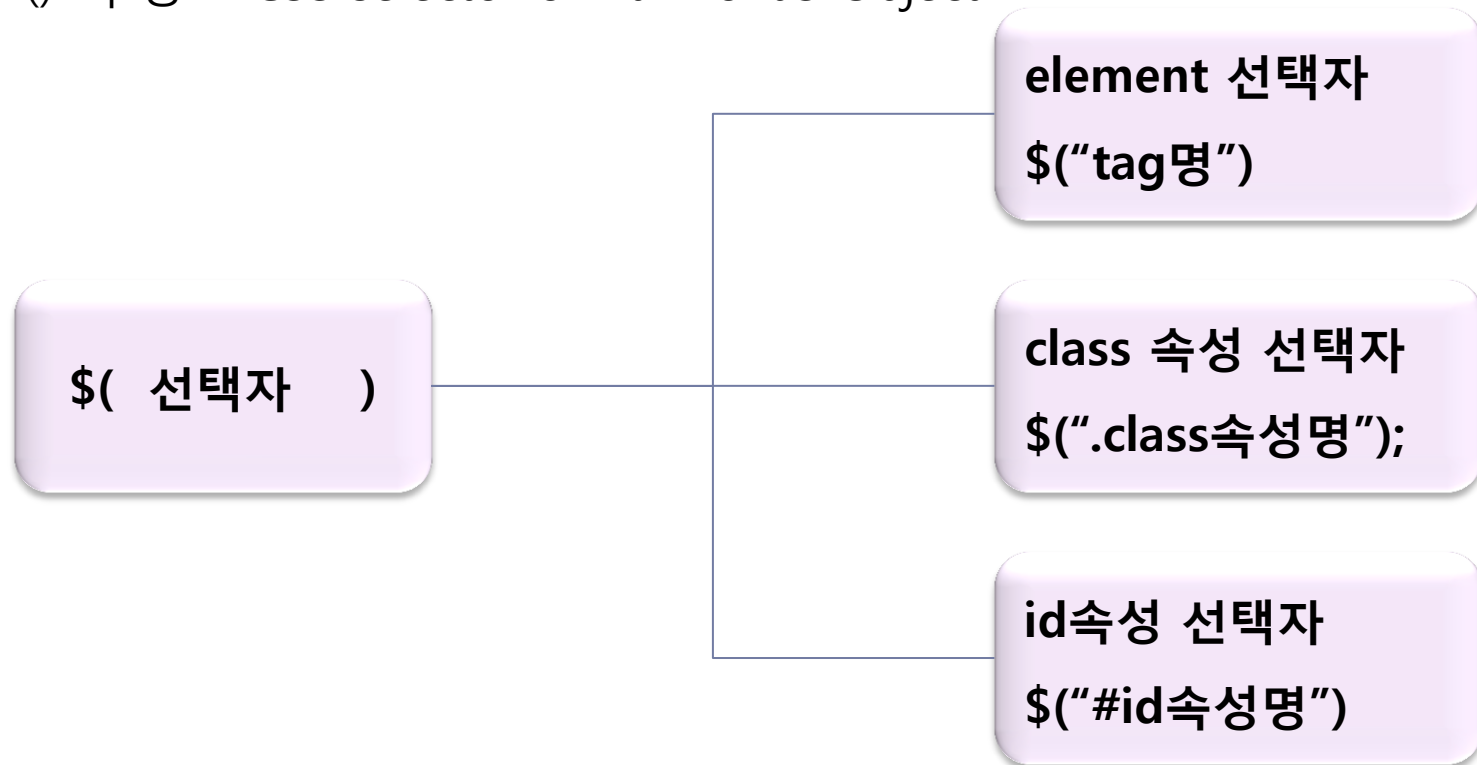
jQuery 문법

1. jQuery()
 1. jQuery함수 or jQuery 레퍼로 jQuery 기능이 담긴 객체로 생성
2. \$()
 1. jQuery() 단축표기
 2. () 구성 = CSS selector or html or JS Object
 3. 주의사항
 1. 하나의 project 에 다양한 toolkit 사용 가능
 2. 발생 가능한 문제 - toolkit표기 동일, 이 경우 명확한 구분이 필요 따라서 \$ 표기 중복되면 jQuery 함수명으로 호출 필수
3. jQuery Ready Event
 1. - DOM이 모두 로딩된 이후 호출되는 진입점 함수

```
$(document).ready(function(){  
    //jQuery 메소드 호출  
});
```

jQuery selectors 문법

1. jQuery를 이용한 노드 검색
 1. `$()`
 2. `()` 구성 = CSS selector or html or JS Object



jQuery 기본 예제

```
9  <h3>* 자바스크립트와 jQuery 기본 문법을 비교하는 초간단 예제 *</h3><br><hr><br>
10 1. 버튼 클릭시 버튼 메시지가 변경됩니다[기본 자바스크립트 문법]<br>
11  <button id="b1" onClick="bClick()">클릭</button><br>
12  <script>
13      function bClick(){
14          document.getElementById("b1").innerText = "자바스크립트";
15      }
16  </script>|
17  <hr><br>
18  <!-- jQuery 함수 단축표기문법-->
19 2. 버튼 클릭시 버튼 메시지가 변경됩니다[jQuery 함수 단축표기문법]<br>
20  <button id="b2">클릭</button>
21  <script src="scripts/jquery-1.9.1.js"></script>
22  <script>
23      $(document).ready(function(){
24          $("#b2").click(function() {
25              $("#b2").html("jQuery 단축표기");
26          });
27      });
28  </script>
29  <hr><br>
30  <!-- jQuery 함수 문법 -->
31 3. 버튼 클릭시 버튼 메시지가 변경됩니다[jQuery 함수 문법]<br>
32  <button id="b3">클릭</button>
33  <script src="scripts/jquery-1.9.1.js"></script>
34  <script>
35      jQuery(document).ready(function(){
36          jQuery("#b3").click(function() {
37              jQuery("#b3").html("jQuery 함수표기");
38          });
39      });
40  </script>
```

jQuery 기본 예제 실행 화면

★ 자바스크립트와 jQuery 기본 문법을 비교하는 초간단 예제 ★

1. 버튼 클릭시 버튼 메시지가 변경됩니다[기본 자바스크립트 문법]

클릭

2. 버튼 클릭시 버튼 메시지가 변경됩니다[jQuery 함수 단축표기문법]

클릭

3. 버튼 클릭시 버튼 메시지가 변경됩니다[jQuery 함수 문법]

클릭

★ 자바스크립트와 jQuery 기본 문법을 비교하는 초간단 예제 ★

1. 버튼 클릭시 버튼 메시지가 변경됩니다[기본 자바스크립트 문법]

자바스크립트

2. 버튼 클릭시 버튼 메시지가 변경됩니다[jQuery 함수 단축표기문법]

jQuery 단축표기

3. 버튼 클릭시 버튼 메시지가 변경됩니다[jQuery 함수 문법]

jQuery 함수표기

jQuery selectors

Selector	Example	Selects
<u>*</u>	<code>\$("*")</code>	All elements
<u>#id</u>	<code>\$("#lastname")</code>	The element with id="lastname"
<u>.class</u>	<code>\$(".intro")</code>	All elements with class="intro"
<u>.class.class</u>	<code>\$(".intro,.demo")</code>	All elements with the class "intro" or "demo"
<u>element</u>	<code>\$("p")</code>	All <p> elements
<u>el1,el2,el3</u>	<code>\$("h1,div,p")</code>	All <h1>, <div> and <p> elements
<u>:first</u>	<code>\$("p:first")</code>	The first <p> element
<u>:last</u>	<code>\$("p:last")</code>	The last <p> element
<u>:even</u>	<code>\$("tr:even")</code>	All even <tr> elements
<u>:odd</u>	<code>\$("tr:odd")</code>	All odd <tr> elements

jQuery selectors

<u>parent > child</u>	<code>\$("#div > p")</code>	All <p> elements that are a direct child of a <div> element
<u>parent descendant</u>	<code>\$("#div p")</code>	All <p> elements that are descendants of a <div> element
<u>element + next</u>	<code>\$("#div + p")</code>	The <p> element that are next to each <div> elements
<u>element ~ siblings</u>	<code>\$("#div ~ p")</code>	All <p> elements that are siblings of a <div> element
<u>:eq(index)</u>	<code>\$("#ul li:eq(3)")</code>	The fourth element in a list (index starts at 0)
<u>:gt(no)</u>	<code>\$("#ul li:gt(3)")</code>	List elements with an index greater than 3
<u>:lt(no)</u>	<code>\$("#ul li:lt(3)")</code>	List elements with an index less than 3
<u>:not(selector)</u>	<code>\$("#input:not(:empty)")</code>	All input elements that are not empty

jQuery selectors

<u>[attribute]</u>	<code>\$("[href]")</code>	All elements with a href attribute
<u>[attribute=value]</u>	<code>\$("[href='default.htm']")</code>	All elements with a href attribute value equal to "default.htm"
<u>[attribute!=value]</u>	<code>\$("[href!='default.htm']")</code>	All elements with a href attribute value not equal to "default.htm"
<u>[attribute\$=value]</u>	<code>\$("[href\$='.jpg']")</code>	All elements with a href attribute value ending with ".jpg"
<u>[attribute =value]</u>	<code>\$("[title ='Tomorrow']")</code>	All elements with a title attribute value equal to 'Tomorrow', or starting with 'Tomorrow' followed by a hyphen
<u>[attribute^=value]</u>	<code>\$("[title^='Tom']")</code>	All elements with a title attribute value starting with "Tom"
<u>[attribute~=value]</u>	<code>\$("[title~='hello']")</code>	All elements with a title attribute value containing the specific word "hello"
<u>[attribute*=value]</u>	<code>\$("[title*='hello']")</code>	All elements with a title attribute value containing the word "hello"

jQuery selectors

<u>:input</u>	<code>\$(":input")</code>	All input elements
<u>:text</u>	<code>\$(":text")</code>	All input elements with type="text"
<u>:password</u>	<code>\$(":password")</code>	All input elements with type="password"
<u>:radio</u>	<code>\$(":radio")</code>	All input elements with type="radio"
<u>:checkbox</u>	<code>\$(":checkbox")</code>	All input elements with type="checkbox"
<u>:submit</u>	<code>\$(":submit")</code>	All input elements with type="submit"
<u>:reset</u>	<code>\$(":reset")</code>	All input elements with type="reset"
<u>:button</u>	<code>\$(":button")</code>	All input elements with type="button"
<u>:image</u>	<code>\$(":image")</code>	All input elements with type="image"
<u>:file</u>	<code>\$(":file")</code>	All input elements with type="file"
<u>:enabled</u>	<code>\$(":enabled")</code>	All enabled input elements
<u>:disabled</u>	<code>\$(":disabled")</code>	All disabled input elements
<u>:selected</u>	<code>\$(":selected")</code>	All selected input elements
<u>:checked</u>	<code>\$(":checked")</code>	All checked input elements

...

jQuery Event Methods

Method	Description
<u>bind()</u>	Attaches event handlers to elements
<u>blur()</u>	Attaches/Triggers the blur event
<u>change()</u>	Attaches/Triggers the change event
<u>click()</u>	Attaches/Triggers the click event
<u>dblclick()</u>	Attaches/Triggers the double click event
<u>delegate()</u>	Attaches a handler to current, or future, specified child elements of the matching elements
<u>die()</u>	Removed in version 1.9. Removes all event handlers added with the live() method
<u>error()</u>	Deprecated in version 1.8. Attaches/Triggers the error event
<u>event.currentTarget</u>	The current DOM element within the event bubbling phase

jQuery Event Methods

<u>focus()</u>	Attaches/Triggers the focus event
<u>focusin()</u>	Attaches an event handler to the focusin event
<u>focusout()</u>	Attaches an event handler to the focusout event
<u>hover()</u>	Attaches two event handlers to the hover event
<u>keydown()</u>	Attaches/Triggers the keydown event
<u>keypress()</u>	Attaches/Triggers the keypress event
<u>keyup()</u>	Attaches/Triggers the keyup event
<u>live()</u>	Removed in version 1.9. Adds one or more event handlers to current, or future, selected elements
<u>load()</u>	Deprecated in version 1.8. Attaches an event handler to the load event
<u>mousedown()</u>	Attaches/Triggers the mousedown event
<u>mouseenter()</u>	Attaches/Triggers the mouseenter event
<u>mouseleave()</u>	Attaches/Triggers the mouseleave event
<u>mousemove()</u>	Attaches/Triggers the mousemove event
<u>mouseout()</u>	Attaches/Triggers the mouseout event
<u>mouseover()</u>	Attaches/Triggers the mouseover event

...

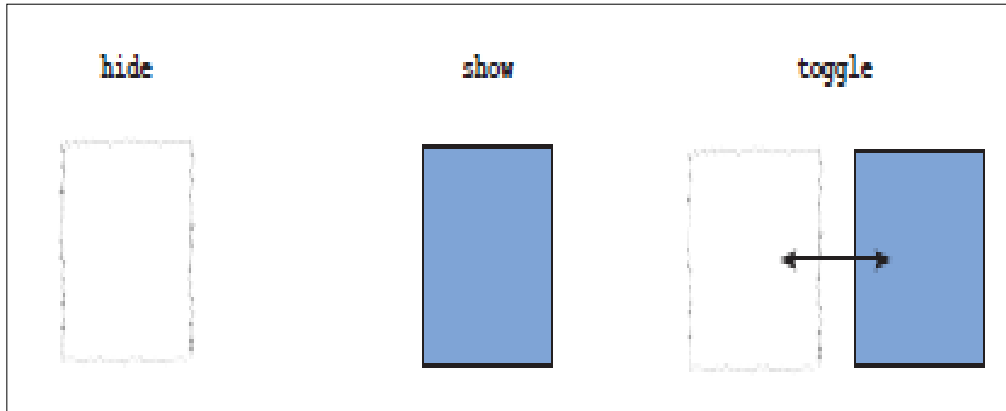
jQuery Effect & 애니메이션

1. CSS 속성을 즉석에서 변경하여 web page의 즉석 변경 효과를 부여
2. 모든 효과가 즉석에서 발생
3. 페이지에서 요소에 전환 효과를 주는 방법
4. 원하는 요소를 보였다 숨겼다 하는 방법
5. 요소를 키웠다, 줄였다 하는 방법

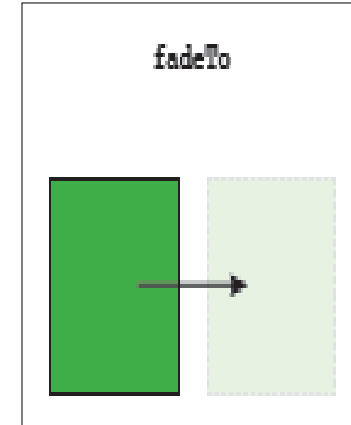
jQuery Effect Methods

Method	Description
<u>animate()</u>	Runs a custom animation on the selected elements
<u>clearQueue()</u>	Removes all remaining queued functions from the selected elements
<u>delay()</u>	Sets a delay for all queued functions on the selected elements
<u>dequeue()</u>	Removes the next function from the queue, and then executes the function
<u>fadeIn()</u>	Fades in the selected elements
<u>fadeOut()</u>	Fades out the selected elements
<u>fadeTo()</u>	Fades in/out the selected elements to a given opacity
<u>fadeToggle()</u>	Toggles between the fadeIn() and fadeOut() methods
<u>finish()</u>	Stops, removes and completes all queued animations for the selected elements
<u>hide()</u>	Hides the selected elements
<u>queue()</u>	Shows the queued functions on the selected elements
<u>show()</u>	Shows the selected elements
<u>slideDown()</u>	Slides-down (shows) the selected elements
<u>slideToggle()</u>	Toggles between the slideUp() and slideDown() methods
<u>slideUp()</u>	Slides-up (hides) the selected elements
<u>stop()</u>	Stops the currently running animation for the selected elements
<u>toggle()</u>	Toggles between the hide() and show() methods

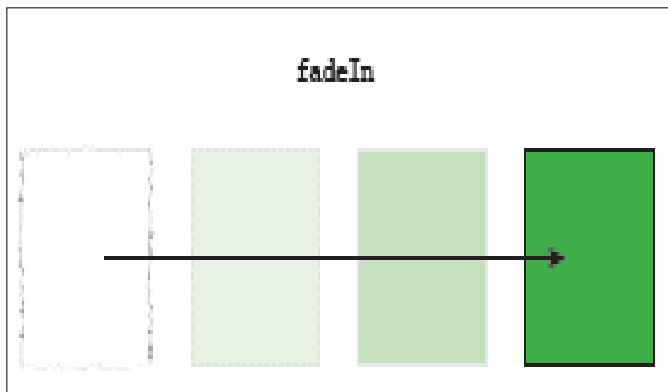
jQuery Effect 및 애니메이션 예 1



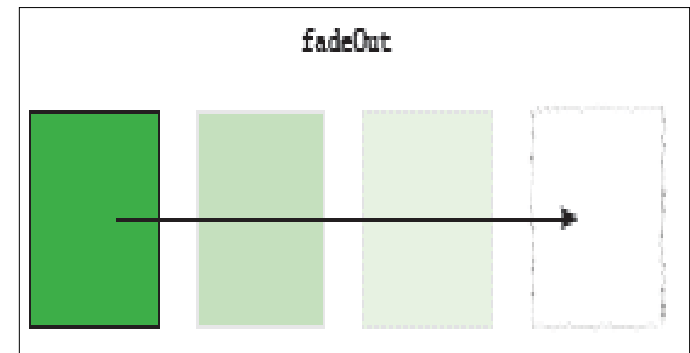
숨겼다 보였다
display속성값 none, display



요소를 지정한 불투명도 퍼센트까지 애니메이션

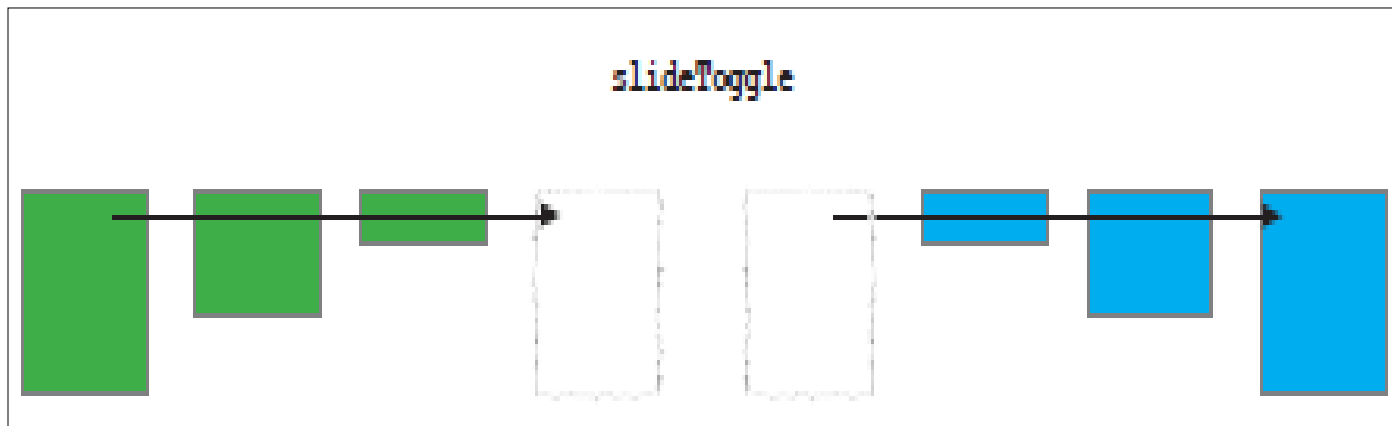
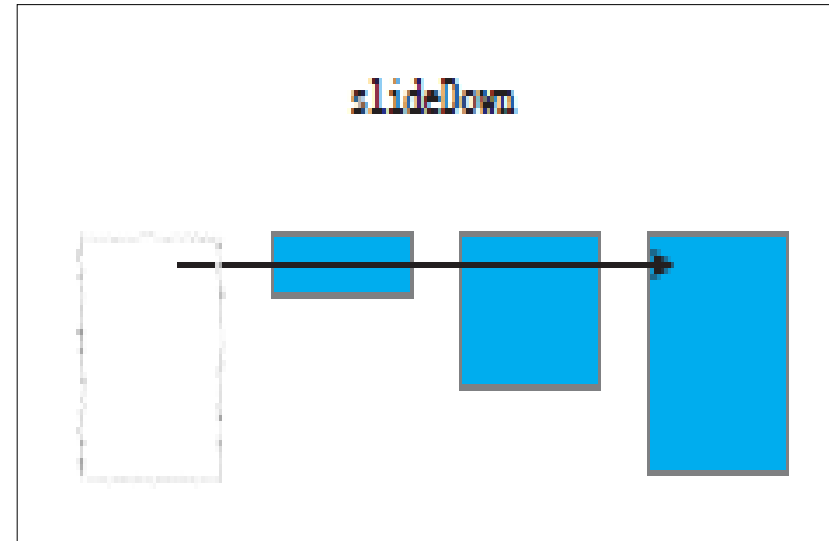
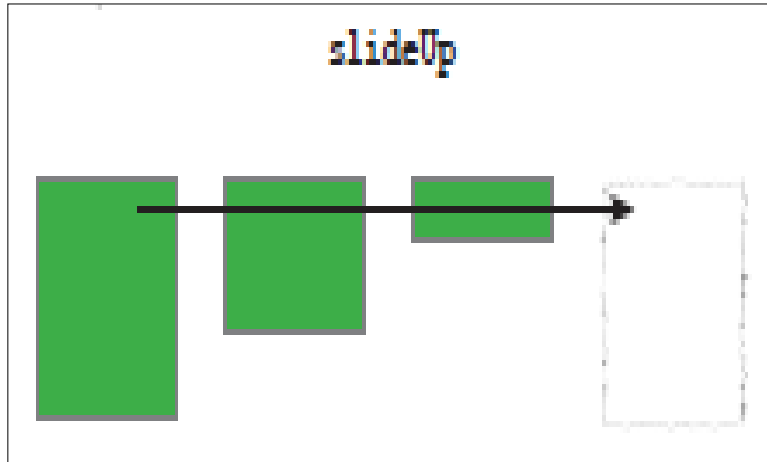


불투명도를 0->100으로



투명도를 100 -> 0으로

jQuery 이펙트 및 애니메이션 예 2



jQuery HTML / CSS Methods

Method	Description
<u>addClass()</u>	Adds one or more class names to selected elements
<u>after()</u>	Inserts content after selected elements
<u>append()</u>	Inserts content at the end of selected elements
<u>appendTo()</u>	Inserts HTML elements at the end of selected elements
<u>attr()</u>	Sets or returns attributes/values of selected elements
<u>before()</u>	Inserts content before selected elements
<u>clone()</u>	Makes a copy of selected elements
<u>css()</u>	Sets or returns one or more style properties for selected elements
<u>detach()</u>	Removes selected elements (keeps data and events)
<u>empty()</u>	Removes all child nodes and content from selected elements
<u>hasClass()</u>	Checks if any of the selected elements have a specified class name
<u>height()</u>	Sets or returns the height of selected elements
<u>html()</u>	Sets or returns the content of selected elements
<u>innerHeight()</u>	Returns the height of an element (includes padding, but not border)
<u>innerWidth()</u>	Returns the width of an element (includes padding, but not border)
<u>insertAfter()</u>	Inserts HTML elements after selected elements

jQuery Traversing Methods

Method	Description
<u>add()</u>	Adds elements to the set of matched elements
<u>addBack()</u>	Adds the previous set of elements to the current set
<u>andSelf()</u>	Deprecated in version 1.8. An alias for addBack()
<u>children()</u>	Returns all direct children of the selected element
<u>closest()</u>	Returns the first ancestor of the selected element
<u>contents()</u>	Returns all direct children of the selected element (including text and comment nodes)
<u>each()</u>	Executes a function for each matched element
<u>end()</u>	Ends the most recent filtering operation in the current chain, and return the set of matched elements to its previous state
<u>eq()</u>	Returns an element with a specific index number of the selected elements
<u>filter()</u>	Reduce the set of matched elements to those that match the selector or pass the function's test
<u>find()</u>	Returns descendant elements of the selected element
<u>first()</u>	Returns the first element of the selected elements
<u>has()</u>	Returns all elements that have one or more elements inside of them
<u>is()</u>	Checks the set of matched elements against a selector/element/jQuery object, and return true if at least one of these elements matches the given arguments
<u>last()</u>	Returns the last element of the selected elements

...

\$(this)

1. this

- 1. 현재 다루고 있는 요소 의미

2. \$(this)

- 1. 현재 다루고 있는 요소에 jQuery 메소드 적용 의미

```
$("#tagClass명").click( function(){  
    $(this).slideUp();  
});
```

jQuery 메소드

jQuery메소드 호출시 함수 실행

jQuery Ajax Programming

jQuery Ajax

1. Java Script로 Ajax를 개발할 경우 대비 jQuery Ajax를 활용하면 쉽고 빠르게 개발 및 처리 가능

Method	Description
<u>\$.ajax()</u>	Performs an async AJAX request
<u>\$.ajaxPrefilter()</u>	Handle custom Ajax options or modify existing options before each request is sent and before they are processed by \$.ajax()
<u>\$.ajaxSetup()</u>	Sets the default values for future AJAX requests
<u>\$.ajaxTransport()</u>	Creates an object that handles the actual transmission of Ajax data
<u>\$.get()</u>	Loads data from a server using an AJAX HTTP GET request
<u>\$.getJSON()</u>	Loads JSON-encoded data from a server using a HTTP GET request
<u>\$.getScript()</u>	Loads (and executes) a JavaScript from a server using an AJAX HTTP GET request
<u>\$.param()</u>	Creates a serialized representation of an array or object (can be used as URL query string for AJAX requests)
<u>\$.post()</u>	Loads data from a server using an AJAX HTTP POST request
<u>ajaxComplete()</u>	Specifies a function to run when the AJAX request completes
<u>ajaxError()</u>	Specifies a function to run when the AJAX request completes with an error
<u>ajaxSend()</u>	Specifies a function to run before the AJAX request is sent
<u>ajaxStart()</u>	Specifies a function to run when the first AJAX request begins
<u>ajaxStop()</u>	Specifies a function to run when all AJAX requests have completed
<u>ajaxSuccess()</u>	Specifies a function to run when an AJAX request completes successfully
<u>load()</u>	Loads data from a server and puts the returned data into the selected element
<u>serialize()</u>	Encodes a set of form elements as a string for submission
<u>serializeArray()</u>	Encodes a set of form elements as an array of names and values

jQuery.ajax() 주요 문법

```
$.ajax({  
    url: " 요청하는 서버단 프로그램명",  
    data: { 서버에 전송할 데이터},  
    type: get 또는 post요청방식,  
    dataType : 응답받을 데이터 타입,  
    success: function( responseData ) {  
        정상 응답 완료시 실행되는 함수  
    } ,  
    error: function( xhr, status ) {  
        비정상 응답시 실행되는 함수  
    }  
});
```

jQuery Ajax 기본 예제

```
20 <h2>* jQuery를 이용한 Ajax 개발 방법 학습 *</h2><br>
21 <script>
22 function ajaxRequest(){
html   $.ajax({
25     url: "responseAjax.jsp",
26     data: {
27         id:$("#id").val()
28     },
29     type: "POST",
30     dataType : "html",
31     success: function( responseData ) {
32         $( "#viewData" ).html(responseData);
33     },
34     error: function( xhr, status ) {
35         alert( "재 요청을 시도하세요" );
36     }
37 }
38 $(document).ready(function(){
39     $("#id").blur(function(){
40         ajaxRequest();
41         $("#id").val("");
42     });
43 });
44 </script>
45 <div id="viewData"></div>
46 <br><hr><br>
47 아이디 : <input type="text" id="id">
```

responseAjax.jsp

```
1 <%@ page language="java"
2     contentType="text/html; charset=EUC-KR"
3     pageEncoding="EUC-KR"%>
4 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
5     "http://www.w3.org/TR/html4/loose.dtd">
6 <html>
7 <head>
8 <meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
9 <title>jQuery를 이용한 Ajax</title>
10 </head>
11 <body>
12     jQuery ajax 함수를 이용한 서버의 응답<br>
13     <font color="blue">안녕하세요 ${param.id}님</font>
14 </body>
15 </html>
```


jQuery Ajax 기본 예제 실행 결과

* jQuery를 이용한 Ajax 개발 방법 학습 *

아이디 :

ajaxMaster 입력후

마우스 이동시 응답 화면

* jQuery를 이용한 Ajax 개발 방법 학습 *

jQuery ajax 함수를 이용한 서버의 응답
안녕하세요 ajaxMaster님

아이디 :