

7강

JSP의 개요

1. 웹프로그래밍의 이해
2. JSP 개요
3. JSP 구조와 요소 (디렉티브, 스크립트릿, 한글처리)
4. JSP 제어문 및 배열출력
5. JSP 구구단 출력
6. 내부객체와 JSP 파라미터 처리
7. 액션 태그의 개요
8. JSP페이지의 모듈화
9. JSP페이지의 흐름제어

1. 웹 프로그래밍의 이해(1/15)

2

□ 웹 프로그래밍의 개요(1/3)

- 개인의 홈 페이지를 만들 경우: HTML태그와 JavaScript만으로도 충분히 작성이 가능.
- 기업의 웹 사이트를 구축할 경우: HTML태그와 JavaScript 만으로는 기업의 방대한 데이터나 쇼핑몰과 같은 실시간으로 수많은 데이터의 변화를 처리하거나 저장하기에는 불가능.
 - 동적으로 변화는 데이터를 처리하고 표시에 문제가 발생.
- 동적으로 변화하는 데이터를 처리하고 표시하기 위해서 개발된 것들이 CGI, ASP, PHP, JSP이다.

1. 웹 프로그래밍의 이해(2/15)

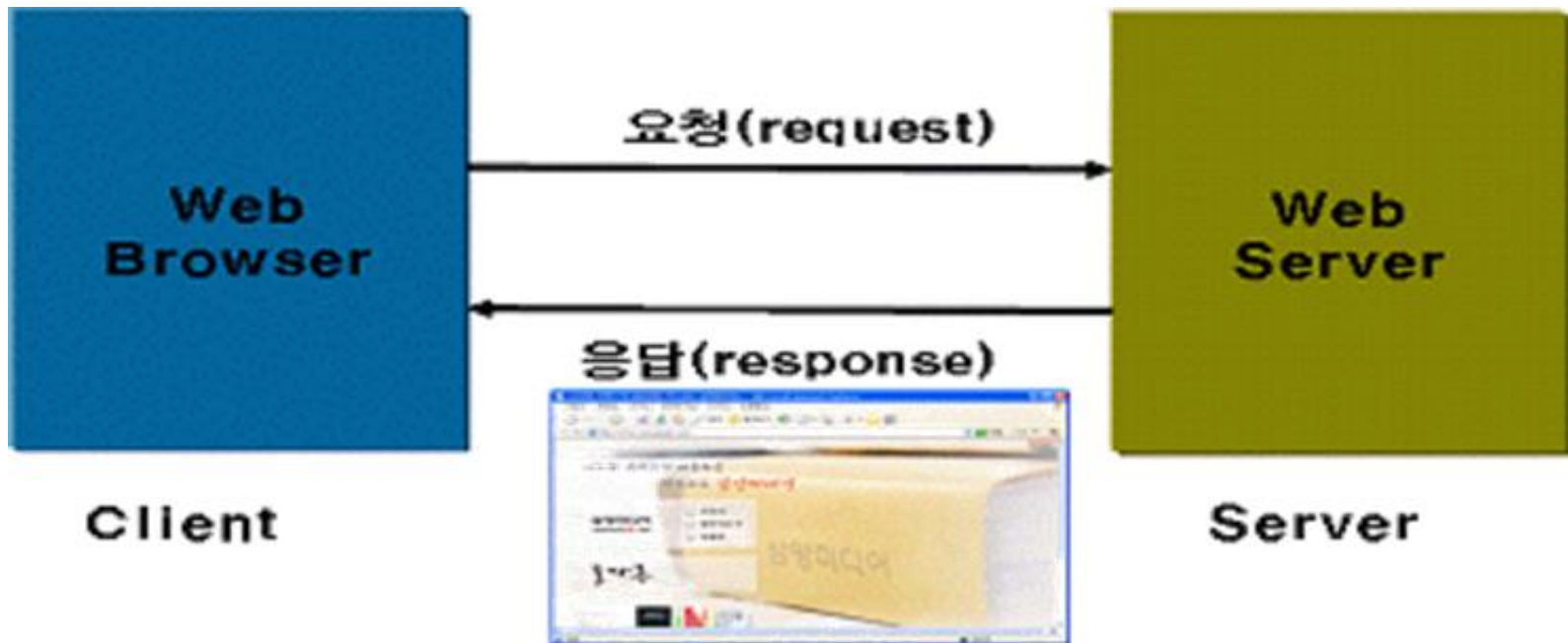
3

- 웹 프로그래밍의 개요(2/3)
 - ▣ 웹 프로그래밍은 기본적으로 클라이언트(Client)/서버(Server) 방식으로 이루어짐.
 - ▣ 클라이언트(웹 브라우저)가 특정페이지를 웹 서버에 요청(Request)하게 되면 웹 서버가 이를 처리한 후 결과를 클라이언트(웹 브라우저)에게 응답(Response)을 하게 되는 구조이다.

1. 웹 프로그래밍의 이해(3/15)

4

□ 웹 프로그래밍의 개요



1. 웹 프로그래밍의 이해(4/15)

5

□ 웹 프로그래밍 언어의 종류(1/4)

▣ CGI(Common Gateway Interface)

- CGI는 Common Gateway Interface의 약자로 사전적인 의미는 월드 와이드 웹(WWW)서버와 백 앤드 프로그램(게이트웨이라고 부른다) 사이에서 정보를 주고받는 데 사용.
- 게이트웨이의 개발 언어로는 UNIX플랫폼(Platform)에서는 문자열 처리가 간단한 펄(Perl), Windows플랫폼(Platform)에서는 비주얼 베이직(Visual Basic) 등이 사용.
- 서비스시 자원을 많이 사용하는 단점이 있음.

1. 웹 프로그래밍의 이해(5/15)

6

□ 웹 프로그래밍 언어의 종류(2/4)

▣ ASP(Active Server Page)

- Microsoft사에서 만들어진 ASP는 비주얼 베이직이라는 언어에서 사용되는 문법들을 사용하여 동적 콘텐츠를 만들어 내기 위한 기술.
- ASP 는 ActiveX란 컴포넌트를 직접 사용할 수 있고, 그런 컴포넌트를 개발하기 위한 기능도 제공.
- 특정 웹 서버와 OS(운영체제)에 동작한다는 단점을 가지고 있으며, Windows플랫폼에서 웹 서버로 IIS(Internet Information Server)만을 사용 .

1. 웹 프로그래밍의 이해(6/15)

7

□ 웹 프로그래밍 언어의 종류(3/4)

▣ PHP(Personal HomePage tools, Professional Hypertext Preprocessor)

- ASP와는 달리 특정영역에서만 동작하지 않고, C 언어의 문법과 유사하기 때문에 기존의 개발자들이 접근이 쉬움.
- 또한 적은 명령어들로서 프로그래밍이 가능하게 되어 있기 때문에 편리성이란 측면에서 많은 이점이 있으나 많은 요구들에 대한 PHP가 지원해주는 기능들이 미약함.(컴포넌트의 문제등)

1. 웹 프로그래밍의 이해(8/15)

8

□ 웹 프로그래밍 언어의 종류(4/4)

▣ Servlet/JSP

- Servlet(Server + Applet) : Sun 사에서 내놓은 기술로서 Java라는 언어를 기반으로 하여 동적인 콘텐츠를 생성하는 기술. Java 코드안에 HTML 태그가 혼재되어 있어서 그 효율성이 떨어짐.
- JSP(Java Server Pages) : JSP 또한 Java 라는 언어를 기반으로 하여 만들어진 것이지만, ASP, PHP처럼 동적 콘텐츠를 생성하기 위해 스크립트 언어 형식으로 프로그램을 작성할 수 있어 개발자에게 쉬운 개발을 할 수 있게 함. 사용자가 직접 태그를 정의해서 사용할 수 있는 사용자 정의 태그를 지정할 수 있는 기능을 갖고 있다.

1. 웹 프로그래밍의 이해(8/15)

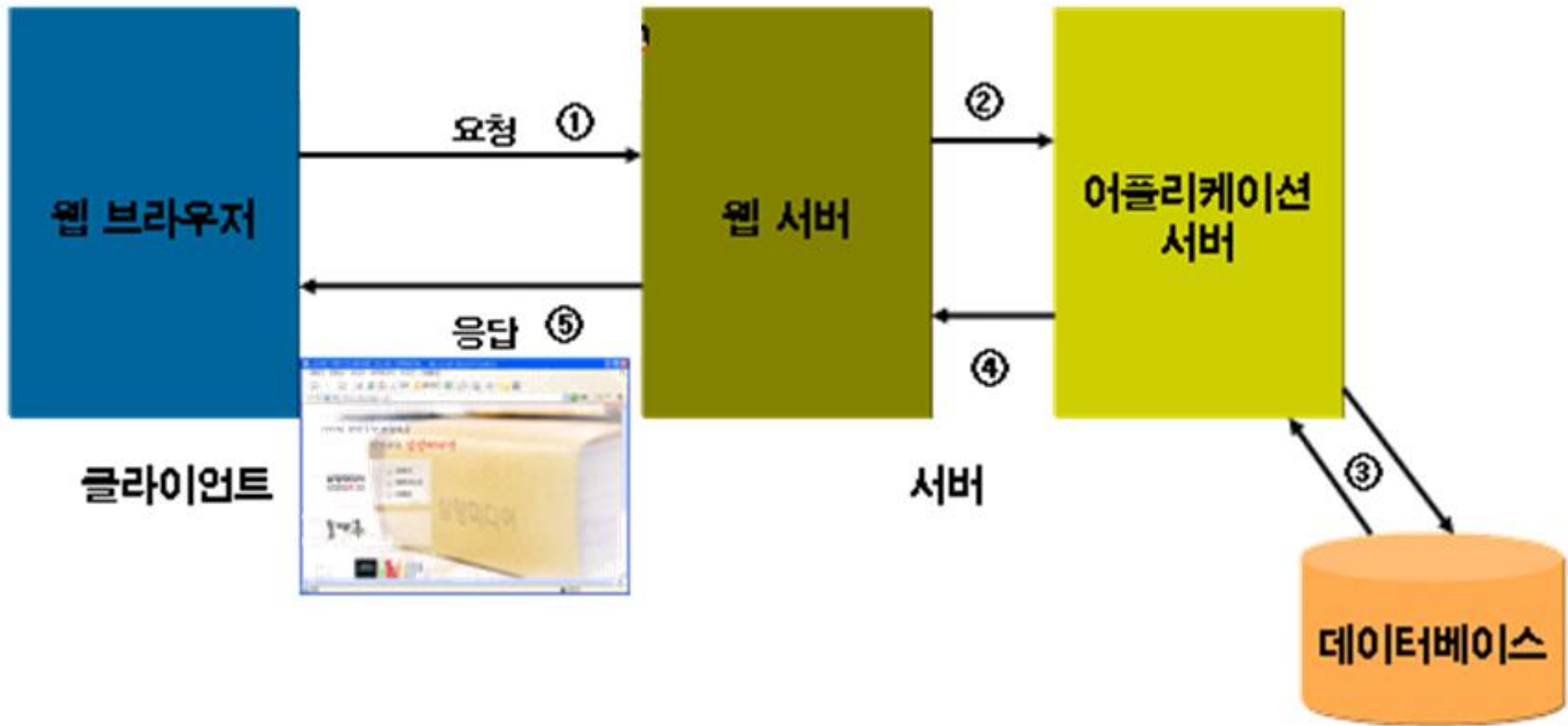
9

- 웹 프로그래밍과 웹 어플리케이션(1/4)
 - ▣ 웹 어플리케이션: 웹을 기반으로 실행되는 프로그램
 - ▣ 웹 프로그래밍과 웹 어플리케이션의 관계: 웹 프로그래밍을 통해 웹 어플리케이션을 구현

1. 웹 프로그래밍의 이해(9/15)

10

□ 웹 프로그래밍과 웹 어플리케이션(2/4)



1. 웹 프로그래밍의 이해(10/15)

11

□ 웹 프로그래밍과 웹 어플리케이션(3/4)

- ① 웹 브라우저가 웹 서버에 어떠한 페이지를 요청하게 되면
- ② 해당 웹 서버는 웹 브라우저의 요청을 받아서 요청된 페이지의 로직 및 데이터베이스와의 연동을 위해 어플리케이션 서버에 이들의 처리를 요청.
- ③ 이때 어플리케이션 서버는 데이터베이스와의 연동이 필요하면 데이터베이스와 데이터의 처리를 수행.
- ④ 로직 및 데이터베이스작업의 처리 결과를 웹 서버에 돌려보냄.
- ⑤ 그러면 웹 서버는 결과를 다시 웹 브라우저에 응답하게 됨.

1. 웹 프로그래밍의 이해(11/15)

12

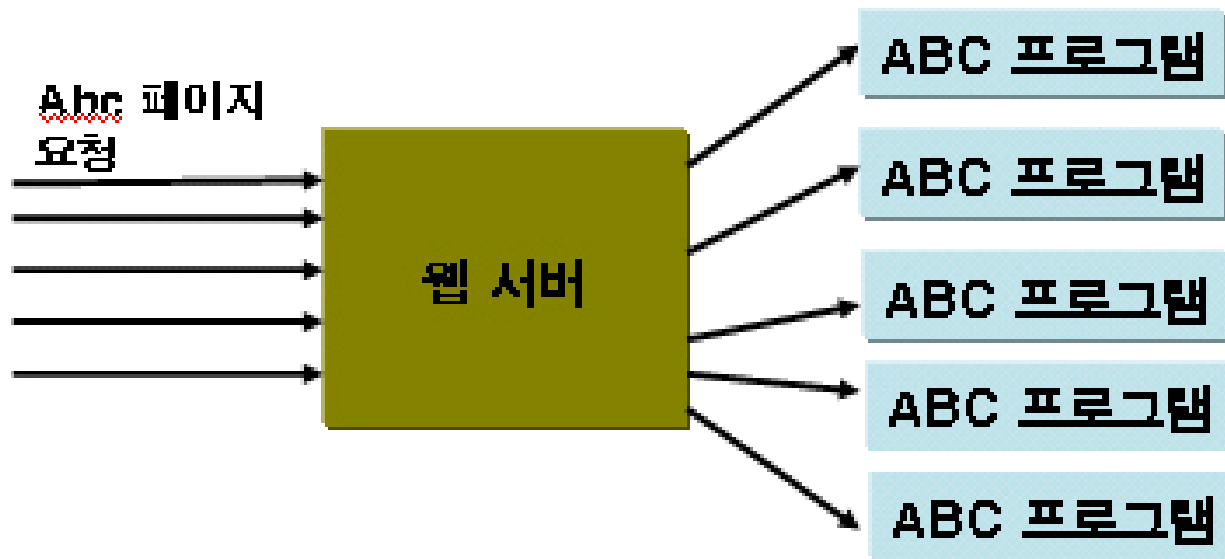
□ 웹 프로그래밍과 웹 어플리케이션(4/4)

웹 어플리케이션의 구성요소	기능
웹 브라우저	웹에서 클라이언트이며, 사용자의 작업창이라 할 수 있다.
웹 서버	웹 브라우저의 요청을 받아들이는 곳으로 작업의 결과도 웹 브라우저에게 응답을 하는 곳이다. 요청된 페이지의 로직 및 데이터베이스와의 연동을 위해 어플리케이션 서버에 이들의 처리를 요청하는 작업을 수행한다.
웹 어플리케이션 서버(WAS)	요청된 페이지의 로직 및 데이터베이스와의 연동을 처리하는 부분이다.
데이터베이스	데이터의 저장소로 웹에서 발생한 데이터는 모두 이곳에 저장된다. 게시판의 글들, 회원의 정보 등등

1. 웹 프로그래밍의 이해(13/15)

13

- CGI방식과 웹 어플리케이션 서버 방식(2/4)
 - ▣ CGI방식



1. 웹 프로그래밍의 이해(14/15)

14

□ CGI방식과 웹 어플리케이션 서버 방식(3/4)

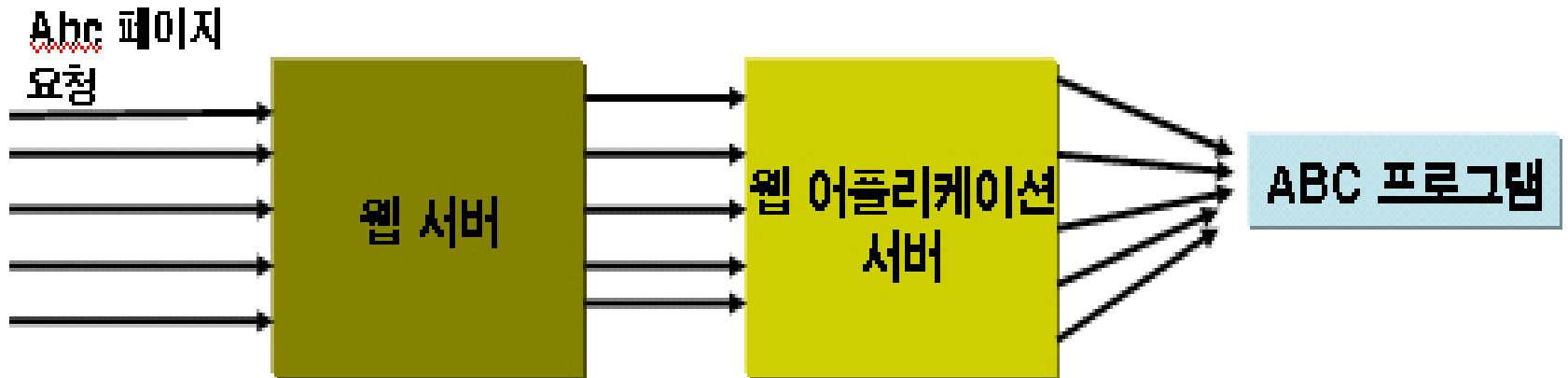
▣ 웹 어플리케이션 서버 방식

- 어플리케이션 서버방식은 웹 서버가 직접 어플리케이션 프로그램을 처리하는 것이 아니라, 웹 어플리케이션 서버에게 처리를 넘겨주고 어플리케이션 서버가 어플리케이션 프로그램을 처리
- 어플리케이션 서버 방식은 여러명의 사용자가 동일한 페이지를 요청하여 같은 어플리케이션 프로그램을 처리할 때 오직 한 개의 프로세스만을 할당하고 사용자의 요청을 스레드(Thread) 방식으로 처리.

1. 웹 프로그래밍의 이해(15/15)

15

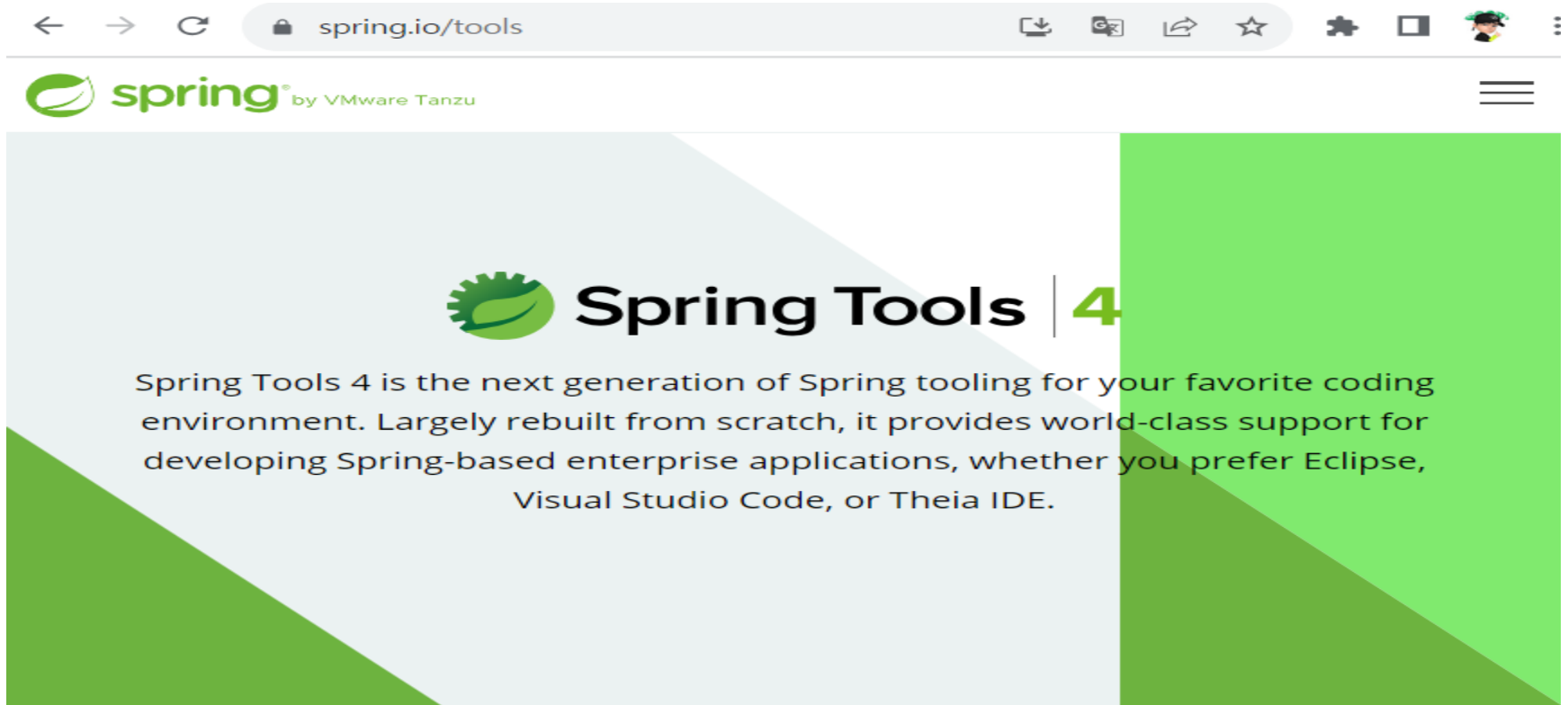
- CGI방식과 웹 어플리케이션 서버 방식(4/4)
 - ▣ 웹 어플리케이션 서버 방식



1. 웹 프로그래밍의 이해 (개발환경)

16

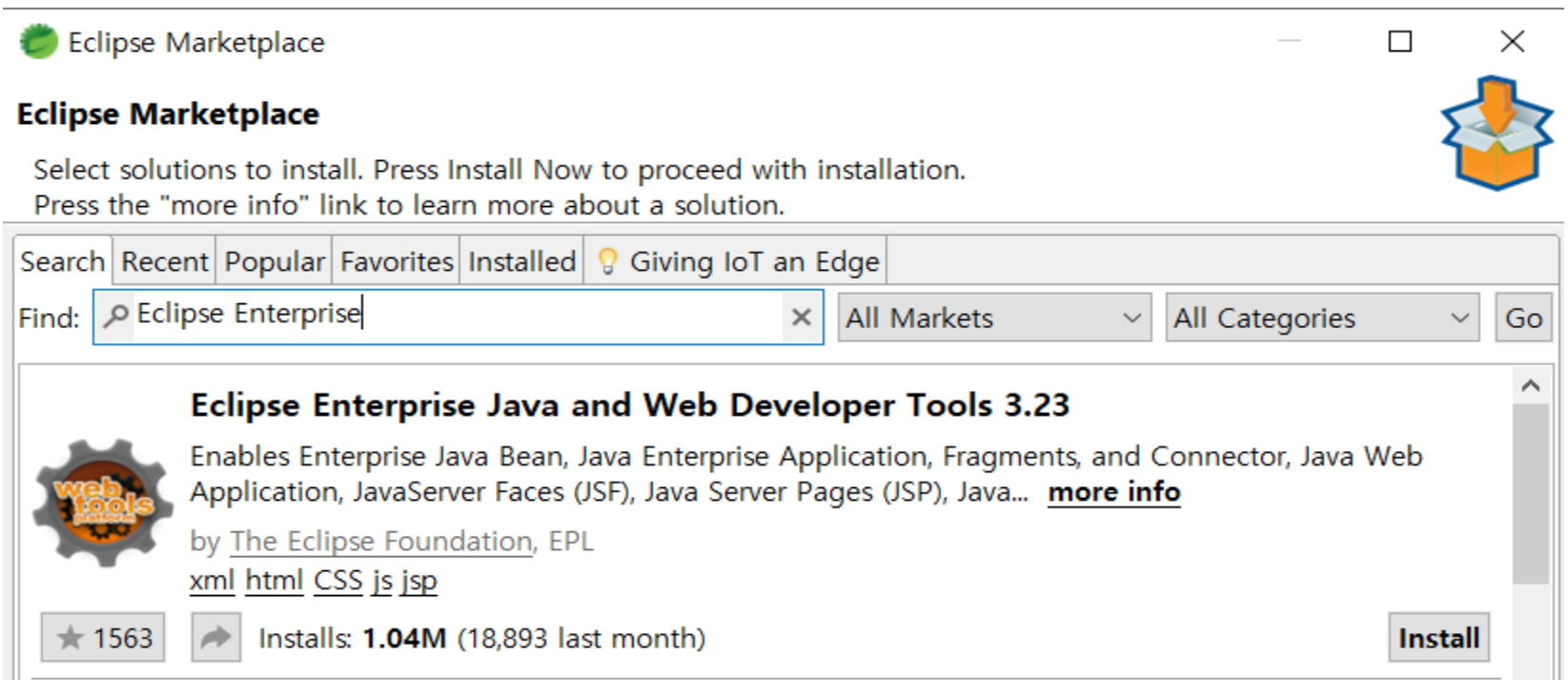
- JDK 설치
 - ▣ JDK 11 버전으로 실습
- 웹개발도구
 - ▣ spring tool suite 4 버전으로 실습



1. 웹 프로그래밍의 이해 (개발환경)

17

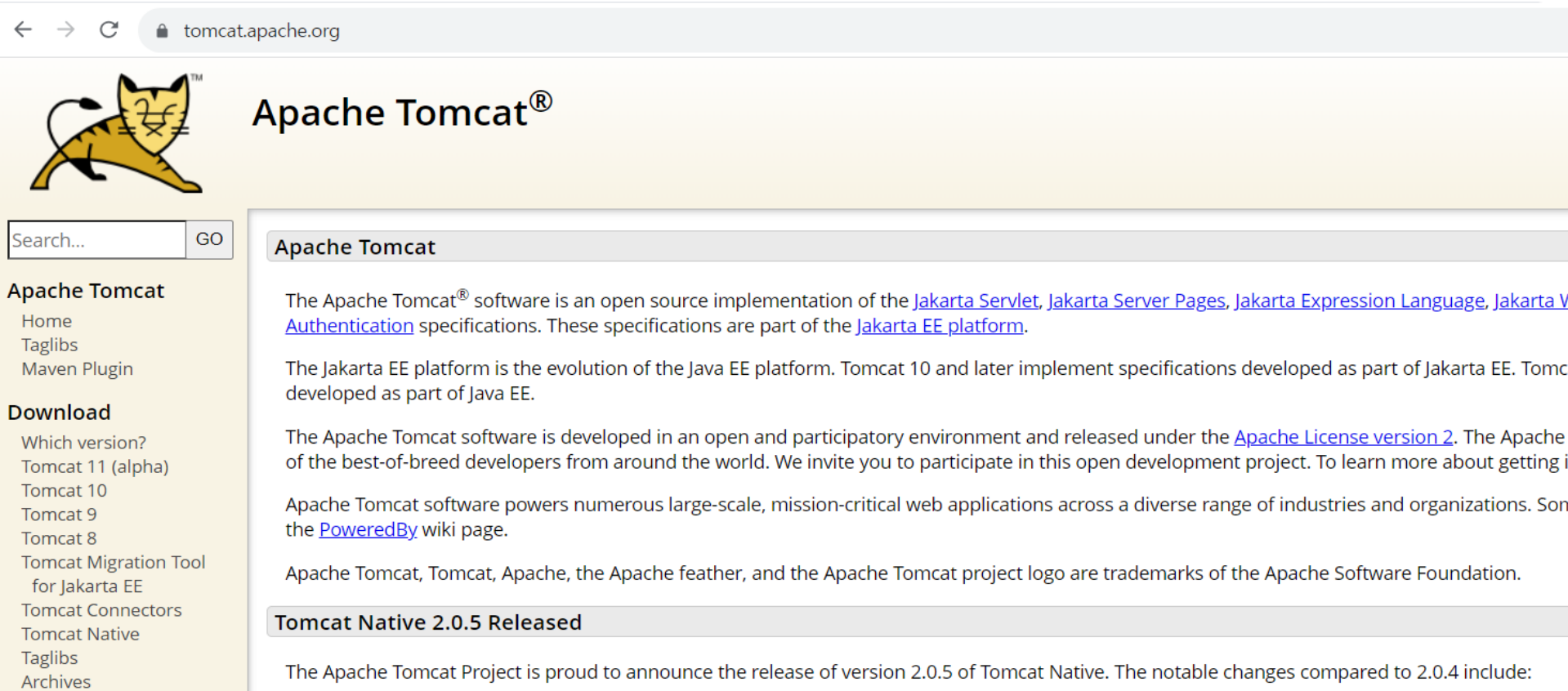
- 웹개발도구
 - ▣ spring tool suite 4 버전으로 실습
- 자바 및 jsp 웹개발을 위해 추가설치
 - ▣ sts4 실행 후 help – eclipse marketplace 클릭
 - ▣ eclipse enterprise 설치



1. 웹 프로그래밍의 이해 (개발환경)


18

- 웹서버 또는 WAS 설치
 - ▣ 톰캣 9 버전으로 실습
- STS4 와 톰캣 연동



The screenshot shows the Apache Tomcat website. The browser address bar displays 'tomcat.apache.org'. The page features the Apache Tomcat logo, a stylized orange cat, and the text 'Apache Tomcat®'. A search bar is located on the left side. The main content area includes a section titled 'Apache Tomcat' with a description of the software as an open source implementation of Jakarta specifications. It also mentions the Jakarta EE platform and the Apache License version 2.0. A sidebar on the left contains links for 'Home', 'Taglibs', 'Maven Plugin', 'Download', and 'Archives'. A 'Download' section lists various versions of Tomcat, including Tomcat 11 (alpha), Tomcat 10, Tomcat 9, and Tomcat 8. A 'PoweredBy' section is also visible, mentioning the 'PoweredBy' wiki page. A 'Tomcat Native 2.0.5 Released' section is at the bottom, announcing the release of version 2.0.5 of Tomcat Native.

← → ↻ 🔒 tomcat.apache.org

 Apache Tomcat®

Search... GO

Apache Tomcat

- Home
- Taglibs
- Maven Plugin

Download

- Which version?
- Tomcat 11 (alpha)
- Tomcat 10
- Tomcat 9
- Tomcat 8
- Tomcat Migration Tool for Jakarta EE
- Tomcat Connectors
- Tomcat Native
- Taglibs
- Archives

Apache Tomcat

The Apache Tomcat® software is an open source implementation of the [Jakarta Servlet](#), [Jakarta Server Pages](#), [Jakarta Expression Language](#), [Jakarta V](#), [Authentication](#) specifications. These specifications are part of the [Jakarta EE platform](#).

The Jakarta EE platform is the evolution of the Java EE platform. Tomcat 10 and later implement specifications developed as part of Jakarta EE. Tomcat 9 was developed as part of Java EE.

The Apache Tomcat software is developed in an open and participatory environment and released under the [Apache License version 2](#). The Apache Tomcat project is 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, see the [PoweredBy](#) wiki page.

Apache Tomcat software powers numerous large-scale, mission-critical web applications across a diverse range of industries and organizations. See the [PoweredBy](#) wiki page.

Apache Tomcat, Tomcat, Apache, the Apache feather, and the Apache Tomcat project logo are trademarks of the Apache Software Foundation.

Tomcat Native 2.0.5 Released

The Apache Tomcat Project is proud to announce the release of version 2.0.5 of Tomcat Native. The notable changes compared to 2.0.4 include:

1. 웹 프로그래밍의 이해 (자바 문법 복습)

19

- 자바 개발 환경
 - ▣ JDK 11
- 변수와 연산자
- 제어문
 - ▣ 조건문과 반복문
- 배열
- 함수
- 클래스
 - ▣ 상속과 인터페이스
- 컬렉션프레임워크
- 제네릭
- 파일입출력

2. JSP의 개요(1/7)

20

□ JSP와 JAVA의 관계(1/2)

- JSP는 Java Server Pages의 약자로 썬 마이크로시스템즈(Sun Microsystems)사의 자바 서블릿(Servlet) 기술을 확장시킨 웹 환경 상에서 100% 순수한 자바만으로 서버 사이드 모듈을 개발하기 위한 기술이다.
- JSP는 DBMS와 같은 백 엔드 서버(Back-end Server)와 연동하여 이들 백 엔드 서버의 데이터를 가공하여 웹 상의 최종적 사용자에게 디스플레이 할 수 있고, 여러 조건에 따라 디스플레이 할 수 있는 내용들을 동적으로 처리할 수 있는 기능을 제공하고 있다.

2. JSP의 개요(2/7)

21

□ JSP와 JAVA의 관계(2/2)

- ▣ JSP는 자바라는 언어를 기반으로 만들어진 것이며 다음과 같은 특징을 가지고 있다.
 - 객체 지향적
 - 플랫폼 독립적
 - 네트워크 지향적
 - 뛰어난 보안성
 - 멀티스레드 기능
 - 친근한 코드

2. JSP의 개요(3/7)

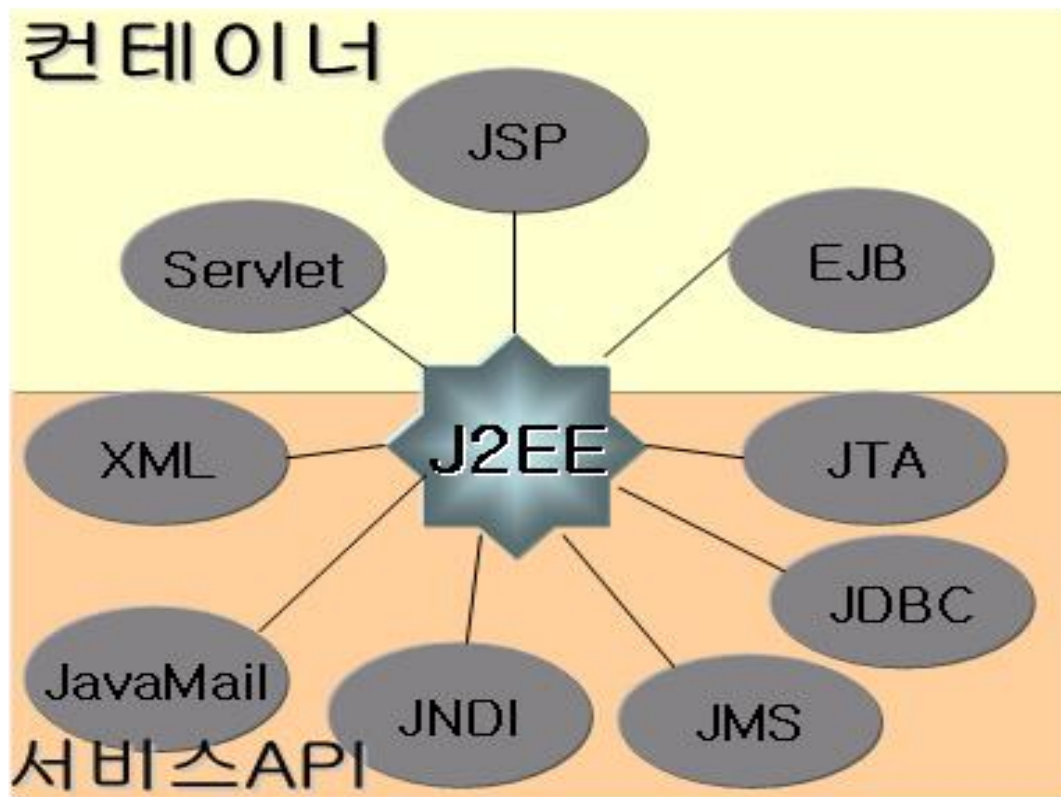
22

- J2EE를 구성하는 기술(1/3)
 - ▣ 자바는 J2SE(Standard Edition), J2EE(Enterprise Edition), J2ME(Micro Edition)으로 나누어져 개발되는데 JSP는 J2EE를 구성하는 기술 중 하나이다.
 - ▣ J2EE는 컨테이너(Container)가 관리하는 컴포넌트(container-managed component) 그룹과 서비스API(Service API) 그룹으로 나뉘어진다.

2. JSP의 개요(4/7)

23

□ J2EE를 구성하는 기술(2/3)



2. JSP의 개요(5/7)

□ J2EE를 구성하는 기술(3/3)

▣ 컨테이너측면과 서비스API측면을 구별하는 이유

- 각 분야마다 전문가들이 개별컴포넌트 하나만 집중해서 개발하면 된다.
- 일선 개발자들은 비즈니스 로직 개발에 집중할 수 있다

2. JSP의 개요(6/7)

25

- 컨테이너(Container)(1/3)
 - ▣ 웹 컨테이너(Web Container): 서블릿과 JSP 에 대한 실행환경을 제공
 - ▣ EJB 컨테이너(EJB Container): Enterprise JavaBean 에 대한 실행환경 제공
 - ▣ 컨테이너를 구성하는 3가지 기술들
 - Servlet
 - JSP(Java Server Page)
 - EJB(Enterprise Java Beans)

2. JSP의 개요(7/7)

26

- 서비스API(Service API) (1/1)
 - ▣ 서비스API(Application Interface)는 실제로 사용하는 각종서비스 환경을 제공
- 서비스API(Service API) 그룹
 - ▣ JDBC 2.0 API:
 - ▣ XML(eXtensible Markup Language):.
 - ▣ JavaMail:
 - ▣ JTA(Java Transaction API):.
 - ▣ JMS(Java Massaging System):
 - ▣ JNDI(Java Naming and Directory Interface):

2. JSP의 개요(7/7)

27

- Hello World
 - ▣ 샘플 – hello.jsp

3. JSP페이지의 디렉티브(1/9)

28

- JSP 페이지의 디렉티브(Directive)는 클라이언트의 요청에 JSP 페이지가 실행이 될 때 필요한 정보를 JSP 컨테이너에게 알리는 역할
 - ▣ 단지 JSP 페이지에 “이렇게 처리를 하세요”라는 지시를 내리는 것
- 디렉티브 태그 안에서 @로 시작하며 3가지 종류가 있다.
 - ▣ page
 - ▣ include
 - ▣ taglib

3. JSP페이지의 디렉티브(2/9)

29

- page 디렉티브(Directive) - `<%@ page%>`
 - ▣ JSP page 디렉티브는 JSP 페이지에서 JSP 컨테이너에게 해당 페이지를 어떻게 처리할 것인가에 대한 페이지 정보를 알려주는데 사용
 - ▣ `<%@ page contentType="text/html; charset=utf-8"%>`
- page 디렉티브(Directive)의 속성 (1/4)
 - ▣ info 속성 - 페이지를 설명해 주는 문자열
 - ▣ **예제 – directiveExample1.jsp**

3. JSP페이지의 디렉티브(3/9)

30

- page 디렉티브(Directive)의 속성(2/4)
 - ▣ language 속성 - JSP 페이지의 스크립트 요소에서 사용할 언어를 지정하는 속성
 - ▣ contentType 속성 - JSP 페이지의 내용이 어떤 형태로 출력을 할 것 인지 MIME 형식으로 브라우저에 알려 주는 역할을 하는 속성
 - ▣ extends 속성 - JSP 페이지가 Servlet 소스로 변환되는 시점에서 자신이 상속 받을 클래스를 지정할 때 사용, 거의 사용되지 않는 속성
 - ▣ import 속성 - 다른 패키지에 있는 클래스를 가져다 쓸 때 사용, import 속성은 page 지시자 중에 유일하게 중복 사용이 가능

3. JSP페이지의 디렉티브(4/9)

31

- page 디렉티브(Directive)의 속성(3/4)
 - ▣ session 속성 - HttpSession을 사용할지 여부를 지정하는 속성 (쿠키와 세션에 자세히 다룸)
 - ▣ buffer 속성 - buffer 속성은 JSP 페이지의 출력 크기를 킬로바이트(KB) 단위로 지정하는 속성이며 기본값은 "8KB" (대부분 기본값으로 충분하다)
 - ▣ autoFlush 속성 - JSP 페이지의 내용들이 브라우저에 출력이 되기 전에 버퍼에 다 찰 경우에 저장되어 있는 내용들을 어떻게 처리 할지를 결정을 하는 속성
 - ▣ **예제-directiveEample2.jsp**

3. JSP페이지의 디렉티브(5/9)

32

- page 디렉티브(Directive)의 속성(4/4)
 - ▣ `errorPage` 속성 - JSP 페이지를 처리하는 도중에 페이지에서 예외가 발생할 경우 자신이 예외를 처리하지 않고 다른 페이지에서 처리하도록 지정할 수 있는 속성
 - ▣ `isErrorPage` 속성 - 현재 JSP 페이지가 에러 처리를 담당하는 페이지인지 아닌지의 여부를 지정할 때 사용되는 속성
 - ▣ `pageEncoding` 속성 - JSP 페이지에서 사용하는 문자(character)의 인코딩을 지정할 때 사용
 - ▣ **예제 - `directiveExample3.jsp`**

3. JSP페이지의 디렉티브(6/9)

33

- include 디렉티브(Directive) - `<%@ include%>` (1/3)
 - ▣ 여러 JSP 페이지에서 공통적으로 포함하는 내용이 있을 때 이러한 내용을 매번 입력하지 않고 별도의 파일을 저장 해 두었다가 JSP 파일에 삽입 가능
 - ▣ 공통적으로 포함하는 내용을 가진 파일을 해당 JSP 페이지 내에 삽입하도록 하는 것이 include 디렉티브
 - ▣ `<%@ include file="로컬URL"%>`

3. JSP페이지의 디렉티브(7/9)

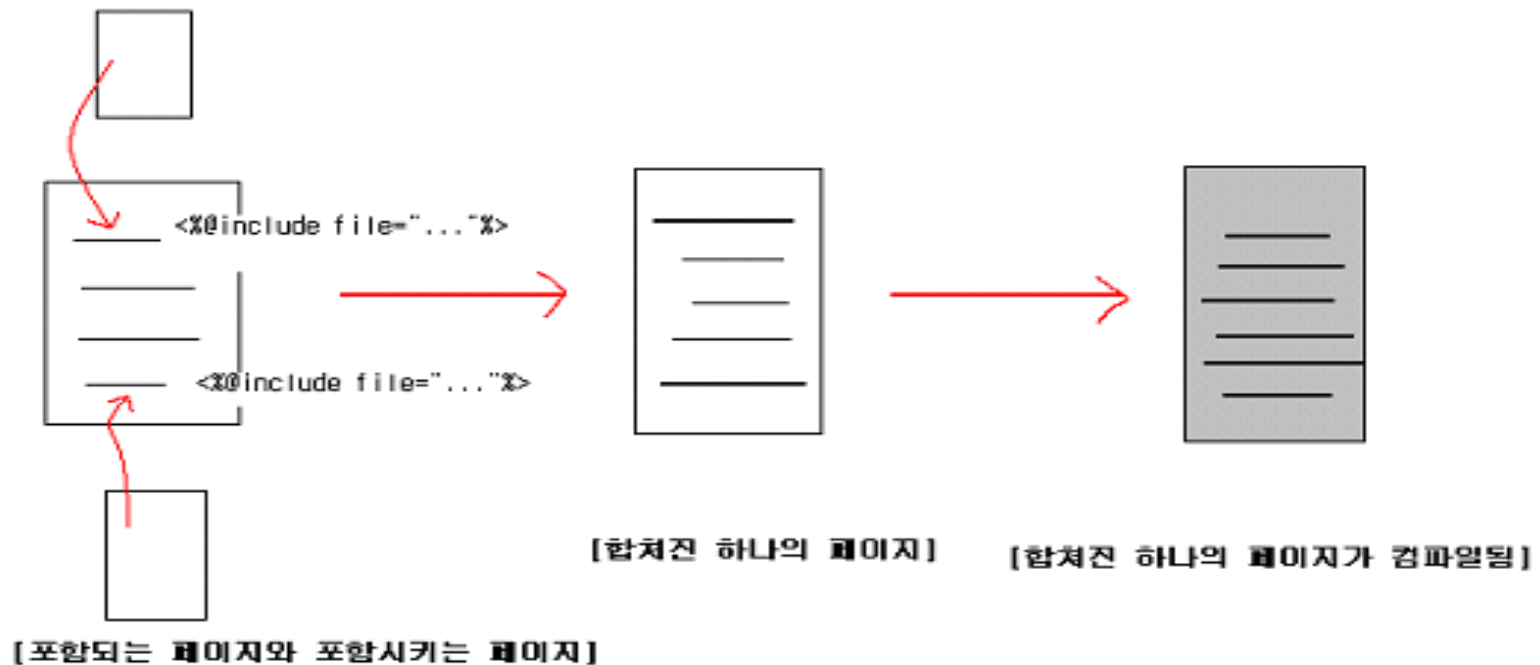
34

- include 디렉티브(Directive) - `<%@ include%>` (2/3)
 - ▣ include 디렉티브의 처리 과정은 정적
 - include 지시자를 사용한 JSP 페이지가 컴파일 되는 과정에서 include 되는 JSP페이지의 소스 내용을 그대로 포함해서 컴파일
 - 즉, 복사 & 붙여넣기 방식으로 두개의 파일이 하나의 파일로 구성된 후 같이 변환되고 컴파일
 - include 디렉티브는 주로 조각 코드를 삽입할 때 사용
 - ▣ 예제 - **directiveExample4.jsp**

3. JSP페이지의 디렉티브(8/9)

35

- include 디렉티브(Directive) - `<%@ include %>` (3/3)
 - ▣ include 디렉티브의 처리 과정



3. JSP페이지의 디렉티브(9/9)

36

- taglib 디렉티브 - `<%@ taglib%>`
 - taglib 디렉티브는 표현언어(EL :Expression Language), JSTL(JSP Standard Tag Library), 커스텀 태그(Custom Tag)를 JSP페이지 내에 사용할 때 사용
 - `<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>`
 - prefix속성과, uri속성의 값을 지정
 - 사용자가 정의한 어떠한 태그라든가의 설정의 정보는 uri속성의 값이 가지고 있고 이것을 해당 페이지 내에서 사용할 때 uri속성의 값이 복잡하므로 prefix속성의 값이 별명과 같은 역할
- 표현 언어, JSTL, 커스텀 태그에서 자세히 다룸

3. JSP페이지의 스크립트요소(1/3)

37

- 스크립트 요소의 이해
 - ▣ JSP에서는 스크립트 요소는 모두 3가지가 있다.
 - 선언문
 - 스크립트릿
 - 표현식
 - ▣ **예제-scriptExample.jsp**
- 선언문(Declaration) - <%! %>
 - ▣ 선언문은 JSP 페이지 내에 안에서 필요한 멤버변수(c언어의 전역 변수라고 생각하면 됨)나 메소드가 필요할 때 쓰이는 요소
 - ▣ **예제-declarationExmaple1.jsp**
 - ▣ **예제-declarationExmaple2.jsp**

3. JSP페이지의 스크립트요소(2/3)

38

□ 스크립트릿(Scriptlet) - `<% %>`

- 스크립트릿(Scriptlet)은 JSP페이지에서 가장 일반적으로 많이 쓰이는 스크립트 요소이고 JSP 페이지가 Servlet으로 변환되고 이 페이지가 호출 될 때 `_jspService` 메소드 안에 선언
- 스크립트릿(Scriptlet)에서 선언된 변수는 지역변수로 선언
- 반드시 초기화를 시켜줄 것

□ 표현식(Expression) - `<%= %>`

- 표현식은 말 그대로 동적인 JSP페이지의 부분을 브라우저로 표현을 하기 위한 코드 부분
- 표현식은 변수의 값도 출력 할 수 있고 메소드의 결과 값도 가져올 수가 있음
 - 스크립트릿 코드 내에서는 `out`이라는 내장객체(*Implicit Objects*)를 통해서 브라우저에 출력도 가능

3. JSP페이지의 스크립트요소(3/3)

39

- 주석(Comment)
 - ▣ HTML 주석 `<!-- -->` : 화면에 표시되지는 않으나, 소스보기로 코드가 표시되고, 주석문안에 실행코드가 있으면 코드가 실행
 - ▣ JSP 주석 `<%-- --%>`
 - ▣ 자바 주석 `//, /**/`
 - JSP 주석 과 자바 주석은 화면에 표시되지도, 실행되지도 않는다.
- 예제-scriptletExmaple1.jsp
- 예제-scriptletExmaple2.jsp

3. JSP페이지에서의 한글처리(1/5)

40

- form에서의 전송방식은 두가지(**GET / POST**)가 존재
- 데이터를 쪽지에 적어서 보내는 것을 GET 이라함
- POST는 보안가방 안에 데이터 쪽지를 담아서 전달
- 비교

Method	속 도	보 안	전송량
GET	빠름	없음	제한적(255 Bytes)
POST	느림	있음	제한 없음

3. JSP페이지에서의 한글처리(2/5)

41

- 웹 브라우저에서 서버로 넘어오는 파라미터 값에 한글이 있는 경우(Get방식) 한글처리
 - ▣ 톰캣홈\conf 폴더에 있는 server.xml 파일에 <Connector>태그의 속성에 `URIEncoding="UTF-8"`문장을 추가.

```
<Connector port="8080" maxHttpHeaderSize="8192"
    maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
    enableLookups="false" redirectPort="8443" acceptCount="100"
    connectionTimeout="20000" disableUploadTimeout="true"
    URIEncoding="UTF-8" />
```

3. JSP페이지에서의 한글처리(3/5)

42

- 웹 브라우저에서 서버로 넘어오는 파라미터 값에 한글이 있는 경우(GET방식) 한글처리

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="UTF-8">
5 <title>Insert title here</title>
6 </head>
7 <body>
8 <h3>한글 폼 테스트 </h3>
9 <form method='GET' action='formPro.jsp'>
10     <input type='text' value='' name='korea'>
11     <input type='submit' value='클릭'>
12 </form>
13 </body>
14 </html>
15
```

```
10 <%
11     String korea = request.getParameter("korea");
12     out.println("한글: " + korea);
13 %>
```

3. JSP페이지에서의 한글처리(4/5)

43

- 서버에서 웹 브라우저에 응답되는 페이지의 화면 출력 시 한글처리
 - ▣ `<%@ page contentType="text/html;charset=utf-8"%>`
- 웹 브라우저에서 서버로 넘어오는 파라미터 값에 한글이 있는 경우(Post방식) 한글처리
 - ▣ `<% request.setCharacterEncoding("utf-8");%>`

3. JSP페이지에서의 한글처리(5/5)

44

- post방식 SetCharacterEncodingFilter.java 를 이용

```
<filter>
  <filter-name>set char encoding</filter-name>
  <filter-class>com.hk.th.util.SetCharacterEncodingFilter</filter-class>

  <init-param>
    <param-name>encoding</param-name>
    <param-value>UTF-8</param-value>
  </init-param>
</filter>

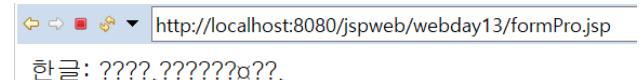
<filter-mapping>
  <filter-name>set char encoding</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```

3. JSP페이지에서의 한글처리(5/5)

45

- 웹 브라우저에서 서버로 넘어오는 파라미터 값에 한글이 있는 경우(POST방식) 한글처리

```
1 <%@ page language="java" contentType="text/html; charset=utf-8"%>
2 <!DOCTYPE html>
3 <html>
4 <head>
5 <meta charset="utf-8">
6 <title>폼전송 GET&POST</title>
7 </head>
8 <body>
9 <h3>폼테스트</h3>
10 <form method="POST" action="formPro.jsp">
11     <input type="text" value="" name="korea">
12     <input type="submit" value="클릭">
13 </form>
14 </body>
15 </html>
```



http://localhost:8080/jspweb/webday13/formPro.jsp
한글: ?????_??????α??,

```
10 <%
11     String korea = request.getParameter("korea");
12     out.println("한글: " + korea);
13 %>
```

4. JSP 제어문 및 배열선언 출력

46

- 표현문
 - ▣ 예제-expressionExmaple1.jsp
 - ▣ 예제-expressionExmaple2.jsp
- 조건분기문
 - ▣ if-else문은 가장 일반적이고 많이 쓰이며 특정한 조건에 의해서 코드 실행의 블록을 조정 할 수 있는 제어문 (폼-ifExample.jsp)

4. JSP 제어문 및 배열선언 출력

47

- 반복문 – for, 반복문 – while
 - ▣ 같은 반복문인데 언제 while문을 쓰고 언제 for문을 쓰는가? for문은 배열과 같이 경계값이 정해진 경우에 주로 사용. 언제나 for문과 배열은 같이 사용. (예제-forExample.jsp)
 - ▣ while문은 그 경계값의 범위를 알 수 없는 경우에 주로 사용. 글 목록과 같이 글이 몇 개인지 확실히 알 수 없는 경우에 주로 사용.
 - ▣ (폼-whileExample.jsp)
- [참고] 자바문법에 대한 기초를 먼저 공부하고 진행

4. JSP 제어문 및 배열선언 출력

48

- 2차원 배열을 선언하고 그림과 같이 숫자를 입력한다
- arrexam.jsp 를 작성하여 배열을 출력하기

그림

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25



출력

```
1 2 3 4 5
6 7 8 9 10
11 12 13 14 15
16 17 18 19 20
21 22 23 24 25
```


5. JSP 구구단 출력 gugu.jsp

49

- `<table>` `<tr>` `<td>` 태그와
- JSP 반복문을 이용하여 출력하시오

구구단

2단	3단	4단	5단	6단	7단	8단	9단
2X1=2	3X1=3	4X1=4	5X1=5	6X1=6	7X1=7	8X1=8	9X1=9
2X2=4	3X2=6	4X2=8	5X2=10	6X2=12	7X2=14	8X2=16	9X2=18
2X3=6	3X3=9	4X3=12	5X3=15	6X3=18	7X3=21	8X3=24	9X3=27
2X4=8	3X4=12	4X4=16	5X4=20	6X4=24	7X4=28	8X4=32	9X4=36
2X5=10	3X5=15	4X5=20	5X5=25	6X5=30	7X5=35	8X5=40	9X5=45
2X6=12	3X6=18	4X6=24	5X6=30	6X6=36	7X6=42	8X6=48	9X6=54
2X7=14	3X7=21	4X7=28	5X7=35	6X7=42	7X7=49	8X7=56	9X7=63
2X8=16	3X8=24	4X8=32	5X8=40	6X8=48	7X8=56	8X8=64	9X8=72
2X9=18	3X9=27	4X9=36	5X9=45	6X9=54	7X9=63	8X9=72	9X9=81

5. JSP 실습 문제 (day14-ex1.jsp)

50

- 2차원 배열을 선언하고 그림과 같이 숫자를 입력한다
- algorithm2.jsp 를 작성하여 아래 출력과 같이 출력하기

그림

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25



출력

```
1 2 3 4 5
10 9 8 7 6
11 12 13 14 15
20 19 18 17 16
21 22 23 24 25
```

6. 내부객체의 개요(1/3)

51

- 내부 객체는 JSP 페이지 내에서 특정한 변수로 호출하고자 하는 변수와 메서드로 접근을 한다.
- JSP 페이지에서 사용하게 되는 특정한 변수가 아무런 선언과 객체 생성 없이 사용 할 수 있는 이유는
- JSP 페이지가 서블릿으로 변환이 될 때 JSP 컨테이너가 자동적으로 제공을 하기 때문이다.

6. 내부객체의 개요(2/3)

52

- 9개의 내부 객체의 사용되는 범주에 따라 4가지 형태로 분류.
 - ▣ JSP 페이지 입출력 관련 기본 객체
 - ▣ JSP 페이지 외부 환경 정보 제공 기본 객체
 - ▣ JSP 페이지 서블릿 관련 기본 객체
 - ▣ JSP 페이지 예외 관련 기본 객체
- 내부 객체의 종류
 - ▣ request, response, out, pageContext, session, application, config, page, exception

6. 내부객체의 개요(3/3)

53

□ 내부 객체가 제공하는 메소드

- ▣ request, session, application, pageContext 내부 객체는 임의의 속성(attribute) 값을 저장하고 읽을 수 있는 메서드를 제공

메서드	리턴 타입	설명
setAttribute (String key, Object value)	void	주어진 key속성의 값을 value로 지정한다.
getAttributeNames()	java.util.Enumeration	모든 속성의 이름을 구한다.
getAttribute (String key)	Object	주어진 key속성의 값을 얻어낸다.
removeAttribute (String key)	void	주어진 key속성의 값을 제거한다.

6. 내부객체(Implicit Object) (1/16)

54

□ request 내부객체(1/3)

- ▣ request 객체는 웹 브라우저에서 JSP 페이지로 전달되는 정보의 모임으로 HTTP 헤더와 HTTP 바디로 구성
- ▣ 웹 컨테이너는 요청된 HTTP 메시지를 통해 HttpServletRequest 객체 타입으로 사용되고 request 객체 명으로 사용

6. 내부객체(Implicit Object) (2/16)

55

- request 내부객체(2/3)
 - ▣ request 내부객체의 요청 메소드
 - ▣ **singlreq.jsp 단일값 받을 때 - radio**
 - ▣ **multireq.jsp 다중값 받을 때 - checkbox**

메서드	설명
String getParameter(name)	이름이 name인 파라미터에 할당된 값을 리턴 하며 지정된 파라미터 값이 없으면 null값을 리턴 한다.
String[] getParameterValues(name)	이름이 name인 파라미터의 모든 값을 String 배열로 리턴 한다. checkbox에서 주로 사용된다.
Enumeration getParameterNames()	요청에 사용된 모든 파라미터 이름을 java.util.Enumeration 타입으로 리턴 한다.

6. 내부객체(Implicit Object) (3/16)

56

□ request 내부객체(3/3)

- ▣ request 객체는 또한 웹 브라우저와 웹 서버의 정보도 가져 올 수가 있다
- ▣ request 내부 객체의 클라이언트 정보 메서드

메서드	설명
String getMethod()	요청에 사용된 요청 방식(GET, POST, PUT)을 리턴 한다.
String getRequestURI()	요청에 사용된 URL로부터 URI을 리턴 한다.
String getQueryString()	요청에 사용된 Query 문장을 리턴 한다.
String getRemoteHost()	클라이언트의 호스트 이름을 리턴 한다.
String getRemoteAddr()	클라이언트의 주소를 리턴 한다.
String getProtocol()	사용 중인 프로토콜을 리턴 한다.
String getServerName()	서버의 도메인 이름을 리턴 한다.
int getServerPort()	서버의 port번호를 리턴 한다.
String getHeader(name)	HTTP 요청 헤더에 지정된 name의 값을 리턴 한다.
String getContextPath()	해당 JSP페이지가 속한 웹 어플리케이션의 컨텍스트 경로를 리턴 한다.

6. 내부객체(Implicit Object) (4/16)

57

□ response 내부객체(1/2)

- ▣ response 객체는 웹 브라우저로 응답할 응답 정보를 가지고 있음.
- ▣ 웹 컨테이너는 요청된 HTTP 메시지를 통해 HttpServletResponse 객체 타입으로 사용되고 response 객체명으로 사용.
- ▣ response 객체는 응답정보와 관련하여 주로 헤더 정보 입력, 리다이렉트 하기등의 기능을 제공.

6. 내부객체(Implicit Object) (5/16)

58

- response 내부객체(2/2)
 - ▣ response 객체의 메서드
 - ▣ **sendredirect.jsp 다른 페이지로 자동 이동**

메서드	설명
void setHeader(name, value)	응답에 포함될 Header를 설정한다.
void setContentType(type)	출력되는 페이지의 contentType 설정한다.
void sendRedirect(url)	지정된 URL로 요청을 재전송한다

6. 내부객체(Implicit Object) (6/16)

59

□ out 내부객체(1/2)

- ▣ out 객체는 JSP 페이지의 결과를 웹 브라우저에 전송해주는 출력 스트림을 나타내며 JSP 페이지가 웹 브라우저에게 보내는 모든 정보는 out 객체로 통해서 전달
- ▣ out 객체는 `java.io.Writer` 클래스를 상속 받은 `javax.servlet.jsp.JspWriter` 클래스 타입의 객체이며 out 객체로 사용
- ▣ 주로 많이 사용되는 메소드는 웹 브라우저에 출력을 하기위한 `println()` 메소드임.
- ▣ out 기본 객체는 출력 버퍼와도 밀접한 관련이 있는데, 사실 JSP 페이지가 사용하는 출력 버퍼는 out 기본 객체가 내부적으로 사용하는 버퍼임.

6. 내부객체(Implicit Object) (7/16)

60

- pageContext 내부객체(1/2)
 - ▣ pageContext 객체는 현재 JSP 페이지의 컨텍스트를 나타내며 pageContext 내부 객체를 통해서 다른 내부 객체에 접근을 할 수가 있음.
 - ▣ pageContext 객체는 javax.servlet.jsp.PageContext 클래스 타입으로 제공 되는 JSP 기본 객체.

6. 내부객체(Implicit Object) (8/16)

61

- out 내부객체(2/2)
 - ▣ out 내부 객체의 메서드 (예제-outex.jsp)

메서드	설명
boolean isAutoFlush()	출력 버퍼가 다 채워진 경우에 자동으로 flush 했을 경우는 true 리턴 그렇지 않은 경우는 false 리턴한다.
int getBufferSize()	출력 버퍼의 전체 크기를 바이트 단위로 리턴한다.
int getRemaining()	출력 버퍼의 남은 양을 바이트 단위로 리턴한다.
void clearBuffer()	현재 출력 버퍼에 저장된 내용을 취소한다(비운다).
String println(string)	주어진 string의 값을 웹 브라우저에 출력을 한다.
void flush()	현재 출력 버퍼에 저장된 내용을 웹 브라우저로 전송하고 버퍼를 비운다.
void close()	출력 버퍼의 내용을 flush하고 스트림을 닫는다.

6. 내부객체(Implicit Object) (9/16)

62

- pageContext 내부객체(2/2)
 - ▣ pageContext 내부 객체의 메서드.

메서드	설명
ServletRequest getRequest()	페이지 요청 정보를 가지고 있는 request 기본 객체를 리턴 한다.
ServletResponse getResopnse()	페이지 요청에 대한 응답 정보를 가지고 있는 response 기본 객체를 리턴 한다.
JspWriter getOut()	페이지 요청에 대한 응답 출력 스트림인 out 기본 객체를 리턴 한다.
HttpSession getSession()	요청한 클라이언트의 세션 정보를 담고 있는 session 기본 객체를 리턴 한다.
ServletContext getServletContext()	페이지에 대한 서블릿 실행 환경 정보를 담고 있는 application 기본 객체를 리턴 한다.
Object getPage()	page 기본 객체를 리턴 한다.
ServletConfig getServletConfig()	페이지의 서블릿 초기 정보 설정 정보를 담고 있는 config 기본 객체를 리턴 한다.
Exception getException()	페이지 실행 중에 발생하는 에러 페이지에 대한 예외정보를 갖고 있는 exception 기본 객체를 리턴 한다.

6. 내부객체(Implicit Object) (10/16)

63

□ session 내부객체(1/2)

- session 객체는 웹 브라우저의 요청에 관한 context 정보의 세션과 관련된 정보(데이터)를 저장하고 관리를 하는 내부 객체
- session 객체는 javax.servlet.http.HttpSession 객체 타입이며 session 객체 명으로 사용.
- session 객체는 웹 브라우저(클라이언트)당 1개가 할당된다. 따라서 주로 회원관리에서 사용자 인증에 관련된 작업을 수행할 때 사용.

6. 내부객체(Implicit Object) (11/16)

64

□ session 내부객체(2/2)

▣ session 내부 객체의 메소드

메서드	설명
String getId()	해당 세션의 세션 ID를 리턴 한다.
long getCreationTime()	세션의 생성된 시간을 리턴 한다.
long getLastAccessedTime()	웹 브라우저의 요청이 마지막으로 시도된 시간을 리턴 한다.
void setMaxInactiveInterval(time)	세션을 유지할 시간을 초단위로 설정한다.
int getMaxInactiveInterval()	setMaxInactiveInterval(time) 로 지정된 값을 리턴 한다. 기본값은 30분으로 지정된다.
boolean isNew()	클라이언트 세션 ID를 할당하지 않은 경우 true 값을 리턴 한다.
void invalidate()	할당된 세션의 속성 값을 모두 제거한다. 주로 세션을 종료시킬 때 사용된다.

6. 내부객체(Implicit Object) (11/16)

65

□ session 내부객체(1/2) (폼-session.jsp)

▣ session 내부 객체의 메소드

메서드	설명
String getId()	해당 세션의 세션 ID를 리턴 한다.
long getCreationTime()	세션의 생성된 시간을 리턴 한다.
long getLastAccessedTime()	웹 브라우저의 요청이 마지막으로 시도된 시간을 리턴 한다.
void setMaxInactiveInterval(time)	세션을 유지할 시간을 초단위로 설정한다.
int getMaxInactiveInterval()	setMaxInactiveInterval(time) 로 지정된 값을 리턴 한다. 기본값은 30분으로 지정된다.
boolean isNew()	클라이언트 세션 ID를 할당하지 않은 경우 true 값을 리턴 한다.
void invalidate()	할당된 세션의 속성 값을 모두 제거한다. 주로 세션을 종료시킬 때 사용된다.

6. 내부객체(Implicit Object) (11/16)

66

□ session 내부객체(2/2)

▣ session 내부 객체의 메소드

리턴 타입	메소드 이름 및 설명
java.lang.Object	getAttribute(java.lang.String name) :name이란 이름에 해당되는 속성값을 Object 타입으로 반환한다.해당되는 이름이 없을 경우에는 null 값을 반환한다.
java.util.Enumeration on	getAttributeNames() : 속성의 이름들을 Enumeration 객체 타입으로 반환한다.
long	getCreationTime() : 1970년 1월 1일 자정을 기준으로 하여 현재 세션이 생성된 시간까지 지난 시간을 계산하여 1/1000초로 반환한다.
java.lang.String	getId() : 세션에 할당된 고유ID를 String 타입으로 반환한다.
int	getMaxInactiveInterval() :현재 생성된 세션을 유지하기 위해 설정된 최대 시간을 정수형으로 반환한다.
void	invalidate() : 현재 생성된 세션을 무효화 시킨다.
void	removeAttribute(java.lang.String name) : name으로 지정한 속성의 값을 제거한다.
void	setAttribute(java.lang.String name, java.lang.Object value) : name으로 지정한 이름에 value 값을 할당한다.
void	setMaxInactiveInterval(int interval) : 세션의 최대 유지시간을 초 단위로 설정한다

6. 내부객체(Implicit Object) (12/16)

67

□ application 내부객체(1/2)

- ▣ application 기본 객체는 서블릿 또는 웹 어플리케이션 환경 정보 (context)를 나타내는 기본 객체.
- ▣ application 객체를 통해서 웹 어플리케이션이 실행되는 서버측의 설정 정보와 자원에 대한 정보를 얻어내거나 어플리케이션이 실행하고 있는 동안에 발생할 수 있는 이벤트 로그와 관련된 기능들을 제공.
- ▣ application 기본 객체는 웹 어플리케이션당 1개의 객체가 생성된다. 따라서 하나의 웹 어플리케이션에서 공유하는 변수로 사용
 - 웹 사이트의 방문자 기록을 카운트할 때 사용된다.

6. 내부객체(Implicit Object) (13/16)

68

- application 내부객체(2/2)
 - ▣ application 객체 관련 메서드 - 웹 어플리케이션의 설정 환경 및 자원 대한 정보를 제공
 - ▣ **예제-applicationExample1.jsp**

메서드	설명
String getServerInfo()	웹 컨테이너의 이름과 버전을 리턴 한다.
String getMimeType(fileName)	지정한 파일의 MIME 타입을 리턴 한다.
String RealPath(path)	지정한 경로를 웹 어플리케이션 시스템상의 경로로 변경하여 리턴 한다.
void log(message)	로그 파일에 message를 기록한다.

6. 내부객체(Implicit Object) (14/16)

69

□ config 내부객체(1/2)

- config 기본 객체는 javax.sevlet.ServletConfig 클래스 타입의 기본 객체.
- ServletConfig객체는 Servlet에게 Servlet을 초기화 하는 동안 참조해야 할 정보를 전해주는 역할을 한다.
 - 서블릿이 초기화 될 때 참조해야 할 다른 여러 정보를 가지고 있다가 전해 준다.
- config 기본 객체는 컨테이너당 1개의 객체가 생성된다. 같은 컨테이너에서 서비스되는 모든 페이지는 같은 객체를 공유.

6. 내부객체(Implicit Object) (15/16)

70

- config 내부객체(2/2)
 - ▣ config 객체 메서드

메서드	설명
Enumeration getInitParameterNames()	모든 초기 파라미터 이름을 리턴 한다.
String getInitParameter(name)	이름이 name인 초기 파라미터의 값을 리턴 한다.
String getServletName()	서블릿의 이름을 리턴 한다.
ServletContext getServletContext()	실행하는 서블릿 ServletContext 객체를 리턴 한다.

4. 내부객체(Implicit Object) (16/16)

71

□ page 내부객체

- ▣ page 기본 객체는 JSP 페이지 그 자체를 나타내는 객체로 JSP 페이지 내에서 page 객체는 this 키워드(this : 자바에서 자기 자신을 가리키는 레퍼런스)로 자기 자신을 참조.

□ exception 내부객체

- ▣ exception 기본 객체는 프로그래머가 JSP 페이지에서 발생한 예외를 처리할 페이지를 지정한 경우 예외 페이지에 전달되는 객체

6. 내부객체의 영역(scope) (1/2)

72

- 웹 어플리케이션은 page, request, session, application 이라는 4개의 영역을 가지고 있음.
- 기본객체의 영역은 객체의 유효기간이라고도 불리며, 객체를 누구와 공유할 것인가를 나타냄.
 - ▣ page영역
 - 한번의 웹 브라우저(클라이언트)의 요청에 대해 하나의 JSP페이지가 호출.
 - 웹 브라우저의 요청이 들어오면 이때 단 한 개의 페이지만 대응.

6. 내부객체의 영역(scope) (2/2)

73

▣ request영역

- 한번의 웹 브라우저(클라이언트)의 요청에 대해 같은 요청을 공유하는 페이지가 대응.
- 웹 브라우저의 한번의 요청에 단지 한 개의 페이지만 요청될 수 있고, 같은 request영역이면 두개의 페이지가 같은 요청을 공유. include 액션 태그, forward 액션 태그를 사용시

▣ session영역

- 하나의 웹 브라우저당 1개의 session객체가 생성.
- 같은 웹 브라우저 내에서는 요청되는 페이지 들은 같은 객체를 공유.

▣ application영역

- 하나의 웹 어플리케이션당 1개의 application 객체가 생성. 같은 웹 어플리케이션에 요청되는 페이지들은 같은 객체를 공유.

7. 액션태그의 개요(1/2)

74

- 액션 태그는 어떤 동작 또는 액션이 일어나는 시점에 페이지와 페이지 사이에 제어를 이동시킬 수도 있음
 - ▣ 브라우저에서 자바 애플릿을 실행시킬 수 있으며, 자바빈도 사용할 수가 있다.
- 액션 태그의 종류
 - ▣ include
 - ▣ forward
 - ▣ plug-in
 - ▣ useBean
 - ▣ setProperty
 - ▣ getProperty

7. 액션태그의 개요(2/2)

75

- 페이지를 모듈화 할 때 <jsp:include> 액션 태그가 사용
- 페이지의 흐름을 제어할 때 <jsp:forward> 액션 태그가 사용
- 자바 빈을 사용할 때(자바 빈 객체 생성시) <jsp:useBean> 액션 태그가 사용
- <jsp:setProperty>, <jsp:getProperty> 액션 태그는 자바 빈의 속성값을 저장하고 읽어 올 때 사용
- <jsp:plug-in> 액션 태그는 애플릿을 사용할 때 애플릿이 웹에서의 서비스가 느린 것 때문에 요즘 잘 사용되지 않음.
 - 애플릿은 JNLP(Java Network Lunching Protocol)로 대체되는 추세

7. JSP페이지의 모듈화(1/8)

- include 액션태그(<jsp:include>)

76

- <jsp:include> 액션 태그는 include 디렉티브(<%@include%>) 와 함께 다른 페이지를 현재 페이지에 포함시킬 수 있는 기능
 - include 디렉티브는 단순히 소스의 내용이 텍스트로 포함이 되지만 <jsp:include> 액션 태그는 포함시킬 페이지의 처리 결과를 포함시킨다는 점이 다름. 포함되는 페이지는 html, jsp, Servlet페이지 모두 가능
 - include 디렉티브는 주로 조각코드를 삽입할 때 사용되고, <jsp:include> 액션 태그는 페이지를 모듈화 할 때 사용된다. 즉, 템플릿 페이지를 작성할 때 사용.

7. JSP페이지의 모듈화(2/8)

- include 액션태그(<jsp:include>)

77

- 사용법
- <jsp:include page="포함될 페이지" flush="false"/>
 - ▣ page속성: 현재 페이지에 결과가 포함될 페이지명
 - ▣ flush 속성: 포함될 페이지로 이동할 때 현재 페이지가 지금까지 출력 버퍼에 저장할 결과를 어떻게 처리 할 것인가를 결정
 - flush 속성의 값은 false로 지정하는 것이 좋음,
 - flush 속성의 값을 true로 지정하면 일단 출력버퍼의 내용을 웹 브라우저에 전송하게 되는데 이때 헤더정보도 같이 전송. 헤더정보가 일단 웹 브라우저에 전송이 되고나면 헤더정보를 추가해도 결과가 반영되지 않음

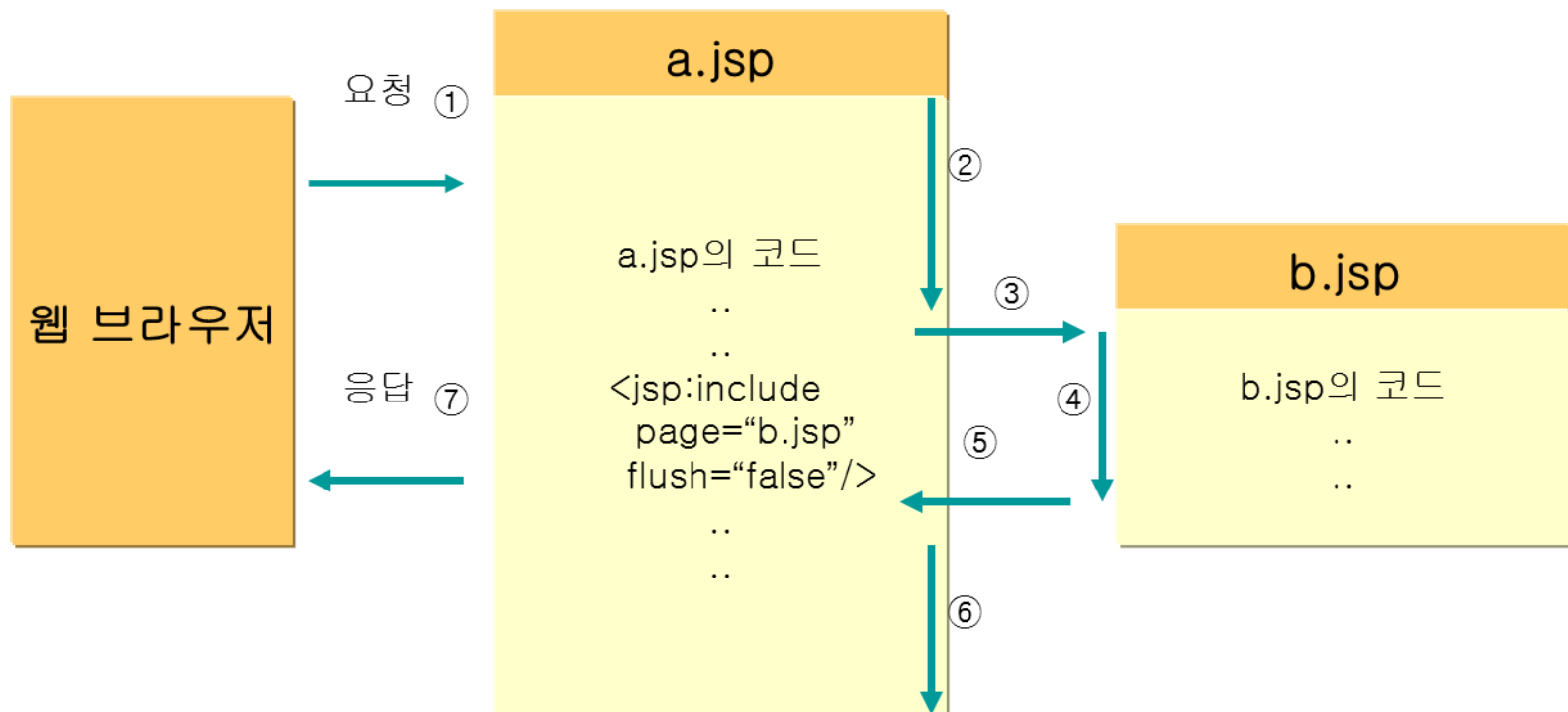
7. JSP페이지의 모듈화(3/8)

- include 액션태그(<jsp:include>)

78

□ include 액션태그의 처리과정)

- ▣ <jsp:include> 액션 태그는 같은 request기본 객체를 공유하므로, a.jsp와 b.jsp는 같은 request기본 객체를 공유한다.



7. JSP페이지의 모듈화(4/8)

- include 액션태그(<jsp:include>)

79

- <jsp:include> 액션 태그에서 포함되는 페이지에 값 전달하기
 - <jsp:include> 액션 태그는 포함되는 JSP 페이지에 값 전달할 수 있음
 - <jsp:include> 액션 태그의 바디(body) 안에 <jsp:param> 액션 태그를 사용
- 예제-includeTag1Form.jsp

7. JSP페이지의 모듈화(5/8)

- include 액션태그(<jsp:include>)

80

- <jsp:param> 액션태그의 속성
 - name속성: 포함되는 JSP 페이지에 전달할 파라미터의 이름을 표시
 - value 속성: 전달할 파라미터의 값을 표시. 이때 value 속성의 값으로 표현식을 사용가능.
 - <jsp:param name="p1" value="<%=var%>" />
- 예제 - includeTag2Form.jsp

```
<jsp:include page="포함되는 페이지" flush="false">  
    <jsp:param name="paramName1" value="var1"/>  
    <jsp:param name="paramName2" value="var2"/>  
</jsp:include>
```


7. JSP페이지의 모듈화(6/8)

- include 액션태그(<jsp:include>)

81

- JSP 페이지의 중복 영역 처리 : JSP 페이지의 모듈화
 - ▣ 중복되는 부분을 따로 페이지로 만들어서 필요할 때 마다 호출해서 사용.
 - ▣ 중복되는 페이지의 호출은 <jsp:include>액션 태그를 사용.
<jsp:include>액션 태그를 페이지의 모듈화라 부름.

7. JSP페이지의 모듈화(7/8)

- include 액션태그(<jsp:include>)

82

- 상단, 좌측메뉴, 하단의 경우 같은 내용을 표시해야 하는 경우가 많고, 주로 중앙의 내용부분의 내용만 계속 바뀌게 됨.
 - ▣ 같은 구조를 계속 유지 하고 있다는 것임.
- 이러한 구조는 <div>태그를 사용해서 작성.



7. JSP페이지의 모듈화(8/8)

- include 액션태그(<jsp:include>)

83

- ▣ 페이지를 모듈화하면 중복되는 페이지만 수정해도 모든 페이지가 수정된 것과 같은 효과를 줌.
 - 사이트의 유지 보수가 쉬워진다.
- ▣ 예제 - `templateTest1.jsp`

9. JSP페이지의 흐름제어(1/4)

-forward 액션태그(<jsp: forward >)

84

- <jsp:forward>액션태그는 다른 페이지로 프로그램의 제어를 이동할 때 사용되는 태그
 - ▣ JSP 페이지 내에 <jsp:forward>액션태그를 만나게 되면, 그전까지 출력버퍼에 저장되어 있던 내용을 제거하고 <jsp:forward>액션태그가 지정하는 페이지로 이동
 - ▣ 사용자가 입력한 값에 따라 여러 페이지로 이동해야 해야 할 경우에 사용하면 좋음.
 - ▣ <jsp:forward>액션태그를 잘 이해하면 모델2(Model2)에 대한 이해가 훨씬 쉬움.

9. JSP페이지의 흐름제어(2/4)

-forward 액션태그(<jsp: forward >)

85

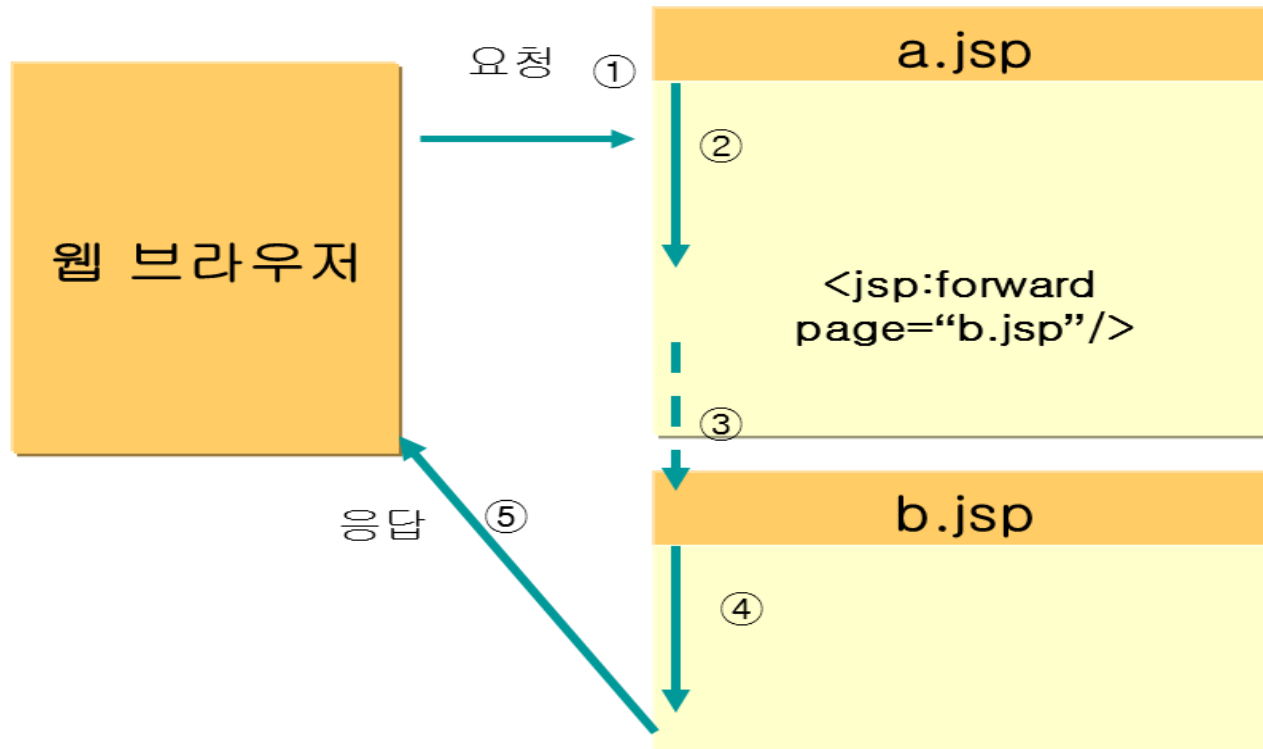
- 사용법
- <jsp:forward page="이동할 페이지명"/>
 - ▣ page속성: 이동할 페이지명을 기술
 - 여기서 page속성의 값인 이동할 페이지명은 웹 어플리케이션 상대경로나 웹 어플리케이션 절대경로로 지정할 수 있고, 표현식도 사용할 수 있음.
 - ▣ 예제 - forwardTag1Form.jsp

9. JSP페이지의 흐름제어(3/4)

-forward 액션태그(<jsp: forward >)

86

- <jsp:forward>액션 태그의 처리과정



9. JSP페이지의 흐름제어(4/4)

-forward 액션태그(<jsp: forward >)

87

- <jsp:forward> 액션 태그에서 포워딩되는 페이지에 값 전달하기 - <jsp:param> 태그의 사용
 - ▣ <jsp:forward> 액션 태그에서 <jsp:param> 태그의 프로그램의 제어가 이동할 페이지에 파라미터 값을 전달할 때 사용 (예제 - forwardexam.jsp)

```
<jsp:forward page="이동할 페이지" >  
    <jsp:param name="paramName1" value="var1"/>  
    <jsp:param name="paramName2" value="var2"/>  
</jsp:forward>
```