

JSP & Framework

목차

Chapter 1. 서블릿과 JSP 개요

Chapter 2. 서블릿 기초

Chapter 3. JSP 기본구조와 문법

Chapter 4. JSP 내장 객체

Chapter 5. JSP 액션 태그

Chapter 6. 자바 빈

Chapter 7. 표현언어와 JSTL

Chapter 8. JDBC

Chapter 9. 파일 업로드

Chapter 10. Ajax

Chapter 11. 모델2 기반의 MVC 패턴

Chapter 12. myBatis 소개 및 설치

Chapter 13. myBatis 활용

Chapter 14. Spring 소개 및 설치

Chapter 15. Spring DI

Chapter 16. Spring AOP

Chapter 17. Spring MVC

Chapter 18. Spring MVC 구현

Chapter1. 서블릿과 JSP 개요

1 서블릿, Servlet

Server + Applet의 합성어

서버에서 실행되는 Applet이라는 의미를 가지고 있으며 자바를 이용하여 웹에서 실행되는 프로그램을 작성하는 기술을 뜻한다. 서블릿은 또한 자바 클래스 형태의 웹 어플리케션을 말하는데 서블릿 클래스를 통해 동작한다.

* 서블릿 클래스

브라우저를 통해 자바 클래스가 실행되도록 하기 위해서는 javax.servlet.http 패키지에서 제공하는 HttpServlet 클래스를 상속받아 구현해야 한다. HttpServlet 클래스를 상속 받아 만든 서브 클래스를 서블릿 클래스라고 한다. 이를 간단히 서블릿이라고 부르기도 한다.

2 JSP, Java Server Page

자바로 서버 페이지를 작성하기 위한 언어

HTML과 JSP 태그(스크립트릿)로 구성되어 화면을 작성하는 데 유리한 웹 프로그래밍 기술
클라이언트가 요청한 페이지를 위한 로직이라 데이터베이스와의 연동을 위해 필요한 것들을 포함한다

브라우저의 요청에 의해 WAS는 내부적으로 서블릿으로 만들어져 실행되게 된다

3 서블릿과 JSP

서블릿과 JSP는 작성방법에 차이가 있을 뿐 결론적으로 동일한 역할을 하게 된다. 초기 자바의 웹 개발은 서블릿을 이용한 개발이었으며 서블릿을 이용한 웹 프로그래밍에서는 자바코드를 기반으로 하기 때문에 HTML 코드를 집어넣기가 상당히 불편하다. 이후 HTML에 자바 코드를 넣을 수 있게 하려는 발상으로 JSP가 나왔으며 JSP는 서블릿과 달리 HTML 코드에 자바코드를 집어넣는 방식으로 개발할 수 있다.

4 개발 환경 구축

웹 어플리케이션 개발을 위해서는 JDK와 톰캣, 이클립스의 설치가 필요하다.

JDK와 이클리스의 설치는 생략한다.

4.1 톰캣 설치하기

웹 어플리케이션을 작성한 후 웹페이지를 브라우저에 띄우기 위해서는 실질적으로 서블릿과 JSP를 구동시키는 WAS가 필요하며 다양한 WAS 중 오픈소스 프로젝트로 개발되어 무료로 제공되는 톰캣을 설치하여 사용하도록 한다.

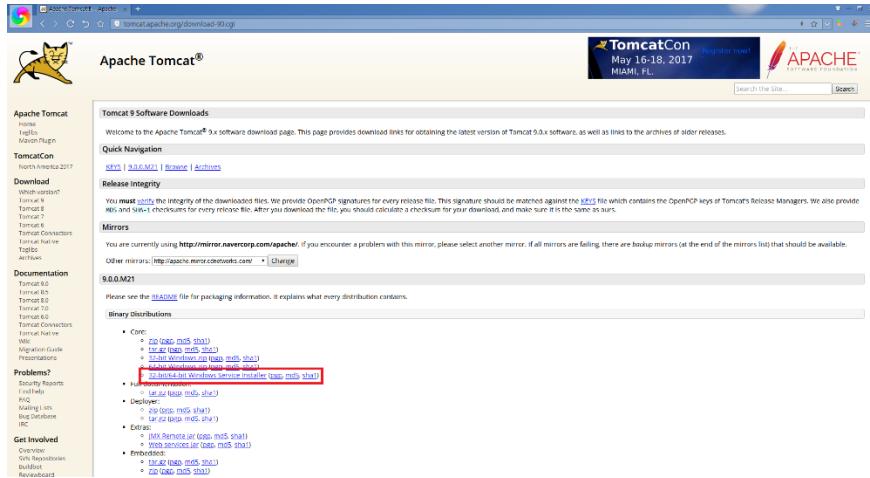
- ① [http://tomcat.apache.org/ 접속](http://tomcat.apache.org/)

The screenshot shows the Apache Tomcat homepage. On the left, there's a sidebar with links like Home, Help, Maven Plugins, TomcatCon North America 2017, Download (with Tomcat 8.0.44 selected), Documentation (with Tomcat 8.0.44 selected), and Problems? (with Security Reports, Feedback, Making JIRA, Migration Guide, and Get Involved). The main content area has sections for Tomcat 8.0.44 Released (May 10, 2017), Tomcat 8.0.44 Released (May 10, 2017), and Tomcat 8.0.15 Released (May 10, 2017). Each release section contains a brief description, a list of changes, and a 'Download' button.

- ② Download – Tomcat9 선택

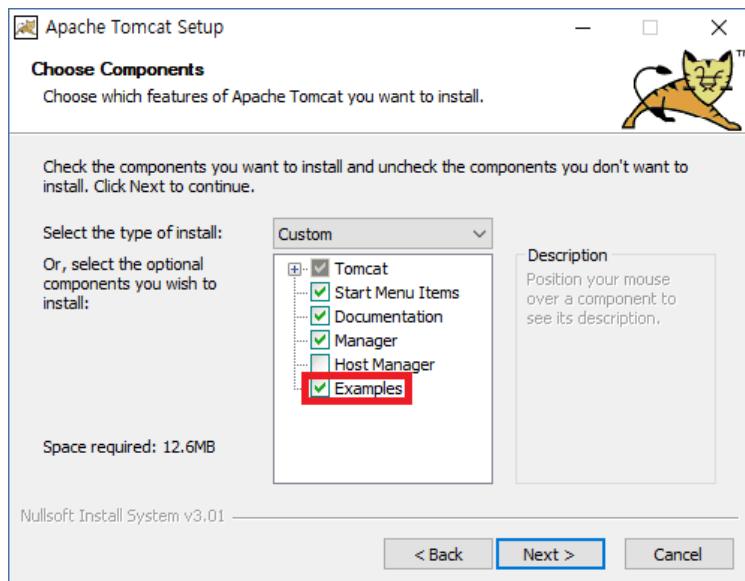
This screenshot is similar to the previous one but shows the Tomcat 9.0.15 release page. The sidebar and the main content area for the Tomcat 8 releases are visible, but the central focus is on the Tomcat 9.0.15 Released section (May 10, 2017), which includes a detailed list of changes and a 'Download' button.

- ③ 32-bit/63-bit Windows Service Installer를 선택해 다운로드



④ 받은 파일을 이용하여 설치 시작

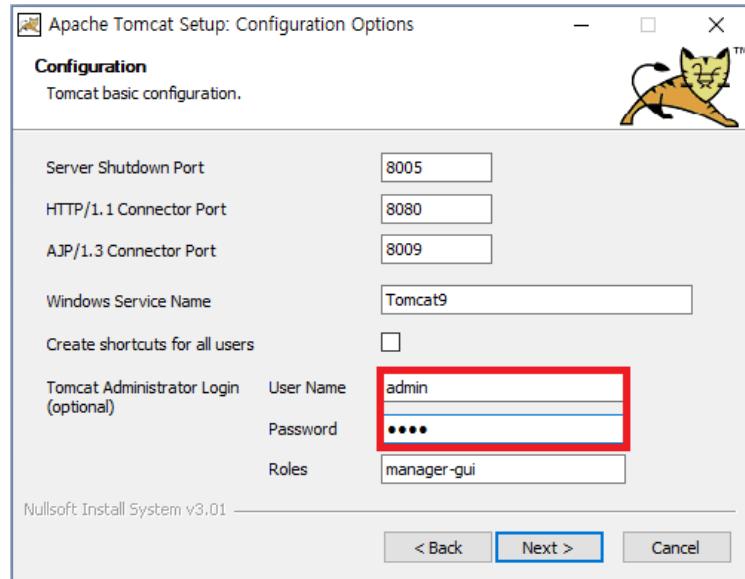
[Next] 버튼을 클릭하여 넘긴 후 환경설정부분에서 Examples를 추가로 체크한 후 [Next] 버튼을 클릭한다. Examples는 후에 확인을 위해 사용할 것이다.



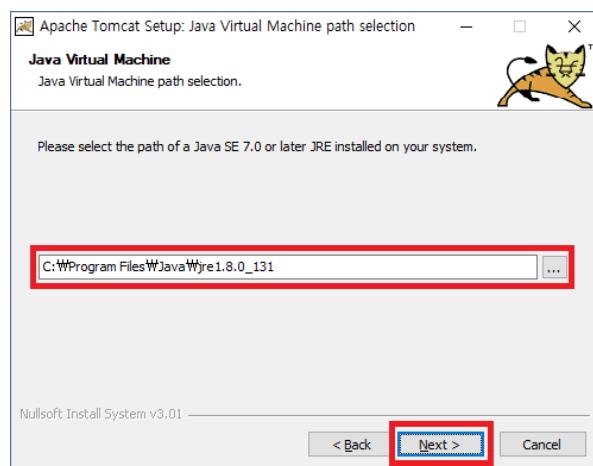
⑤ 환경 설정 중 Administrator Login 이름과 패스워드를 설정한다.

User Name : admin

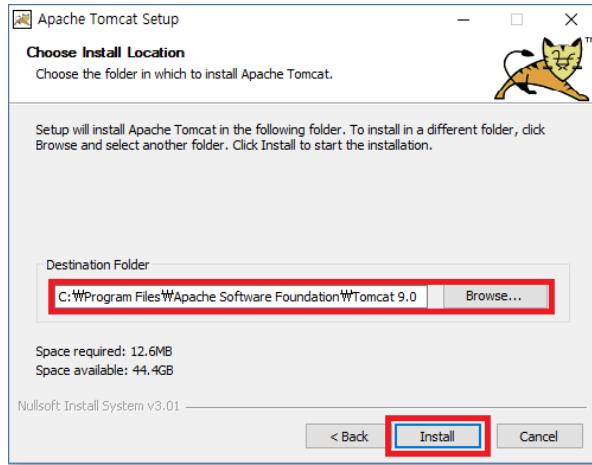
Password : 1234



⑥ JRE 설치 경로를 입력한 후 [Next]



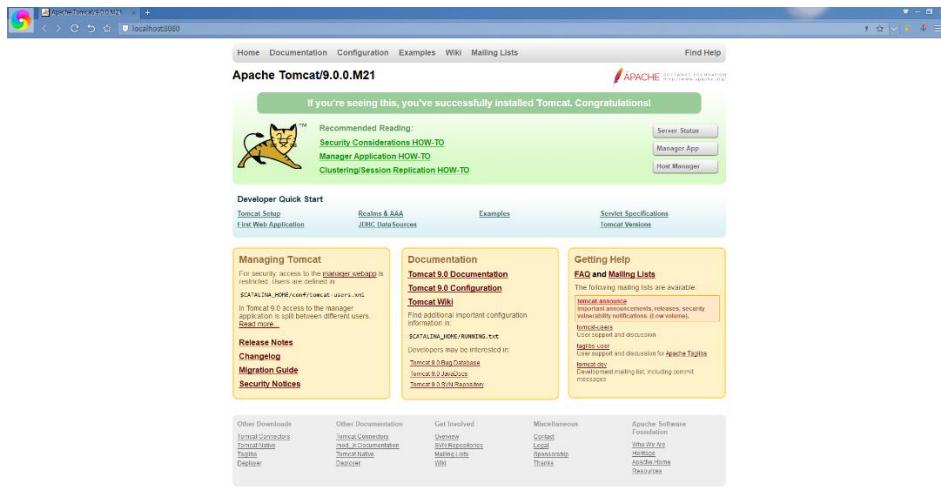
⑦ 톰캣이 설치될 경로를 지정한 후 [Install]



- ⑧ 톰캣 설치가 완료되면 바로 실행이 되며 트레이 아이콘을 통해 확인할 수 있다

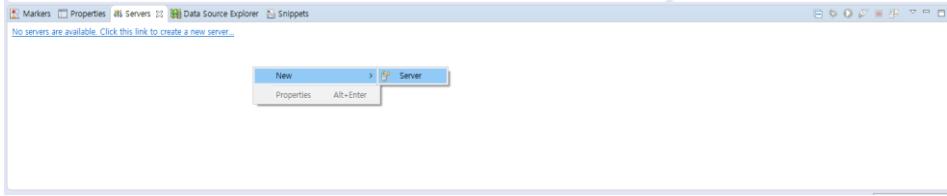


- ⑨ 정상적으로 설치가 완료되었다면 브라우저에 <http://localhost:8080> 을 입력하여 접속하면 톰캣 시작 페이지를 확인할 수 있다

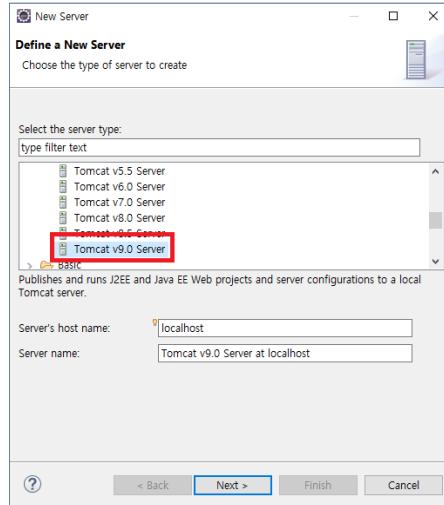


4.2 이클립스에 톰캣 연동하기

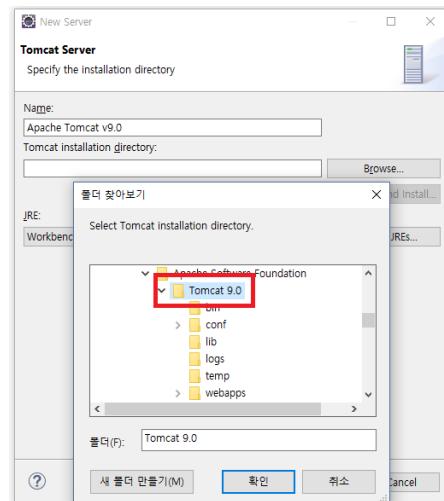
- ① 이클립스의 하단부에 [Servers] 탭을 선택하여 우클릭 메뉴를 통해 [New – Server]를 선택



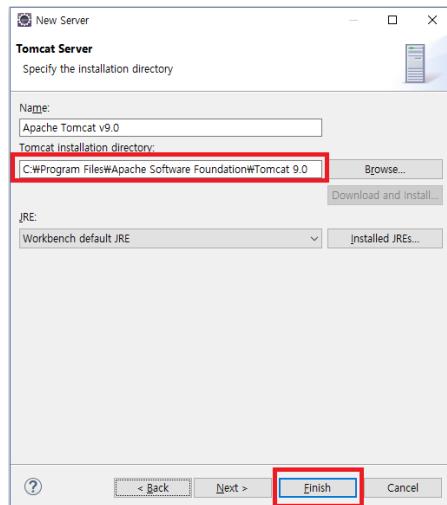
- ② [New Server] 창에서 [Define a New Server]에 설치한 버전과 맞는 버전의 톰캣서버를 선택한 후 [Next]



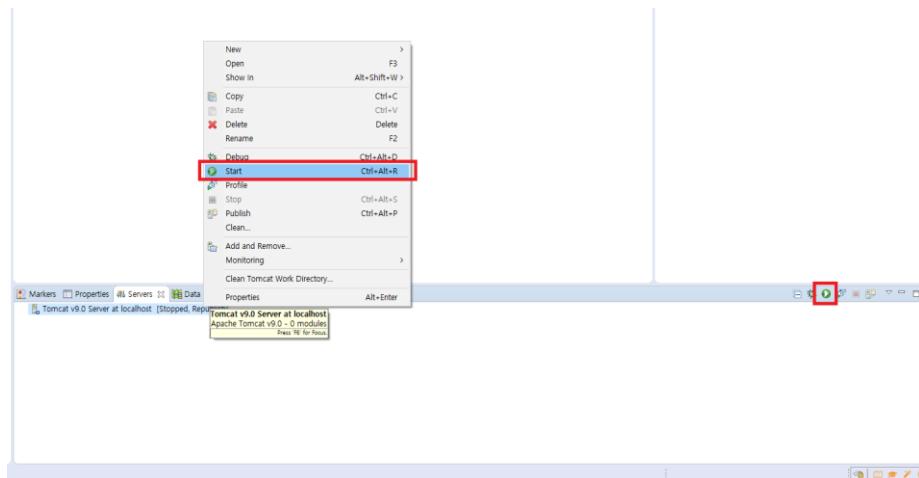
- ③ [Tomcat installation directory] 부분에 톰캣을 설치한 경로를 지정
(기본 경로 : C:\Program Files\Apache Software Foundation\Tomcat 9.0)



④ 경로가 잘 지정되었는지 확인 후 [Finish]



⑤ [Servers] 탭에 추가된 서버를 우클릭하여 [Start]하거나 아이콘을 이용해 서버를 시작시킵니다

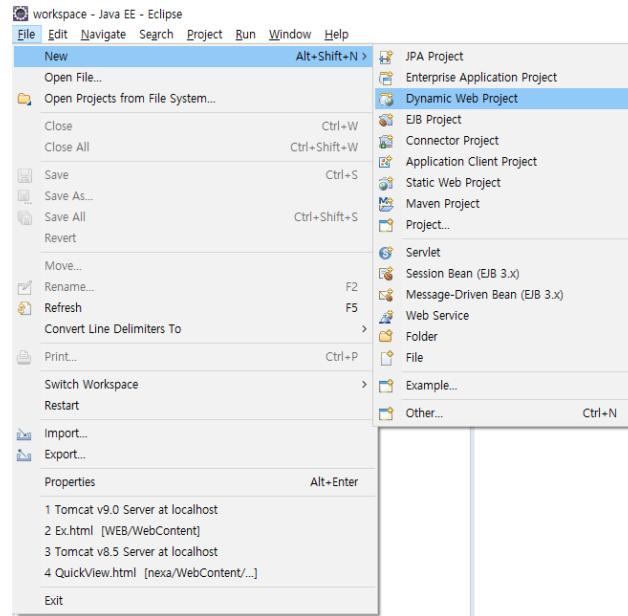


Chapter2. 서블릿 기초

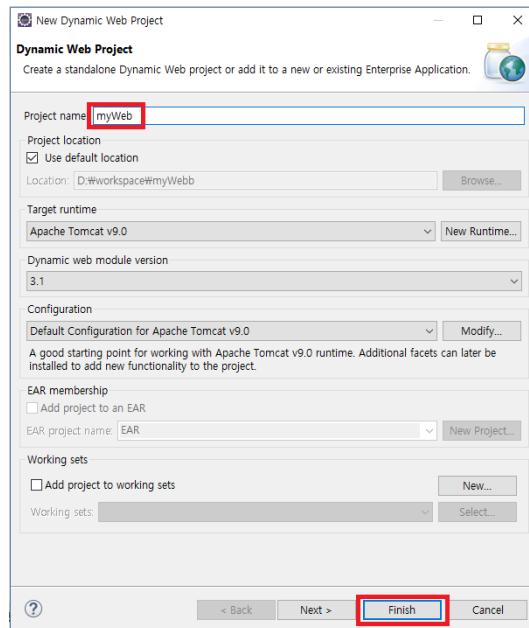
1 기본 서블릿 프로그램 만들기

우선 서블릿을 이용해 기본적인 페이지를 확인할 수 있는 프로그램을 작성해 보자

- ① 이클립스에서 [File – New – Dynamic Web Project 선택]



- ② [Project name]을 myWeb으로 입력하여 프로젝트 생성

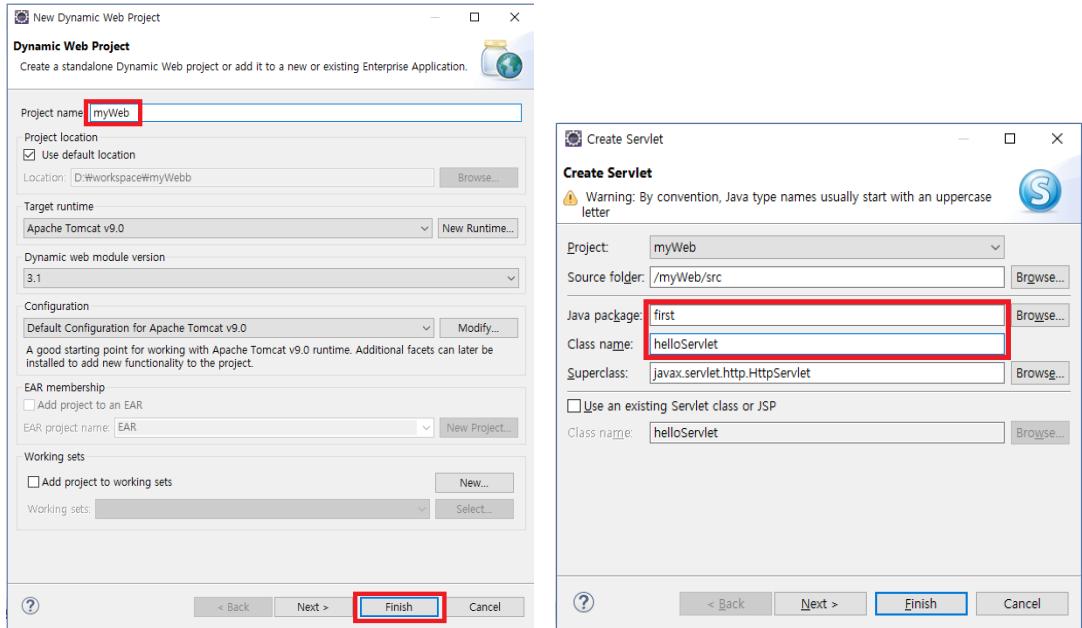


③ 프로젝트 생성 후 [File – New – Servlet] 선택하여

package를 first

class를 helloServlet

으로 지정하고 [Finish]



④ myWeb 프로젝트 내에 Java Resources – src 디렉토리에 first 패키지와 helloServlet.java 서블릿 파일이 생성된 것을 확인할 수 있다.

- helloServlet.java 다음과 같이 작성한 후

브라우저를 통해 <http://localhost:8080/myWeb/helloServlet> 으로 접속

helloServlet.java

```
01 package first;  
02  
03 import java.io.IOException;  
04 import java.io.PrintWriter;  
05  
06 import javax.servlet.ServletException;  
07 import javax.servlet.annotation.WebServlet;
```

```
08 import javax.servlet.http.HttpServlet;
09 import javax.servlet.http.HttpServletRequest;
10 import javax.servlet.http.HttpServletResponse;
11
12 @WebServlet("/helloServlet")
13 public class helloServlet extends HttpServlet {
14     private static final long serialVersionUID = 1L;
15
16     protected void doGet(
17         HttpServletRequest request,
18         HttpServletResponse response
19     ) throws ServletException, IOException {
20
21         response.setContentType("text/html;charset=euc-kr");
22         PrintWriter out = response.getWriter();
23         out.println("<html>");
24         out.println("<body>");
25         out.println("My First Servlet Program");
26         out.println("<br>");
27         out.println("</body>");
28         out.println("</html>");
29
30     }
31 }
```

- 브라우저화면에서 다음과 같이 결과를 확인할 수 있다



My First Servlet Program

- 서블릿은 클라이언트(브라우저)의 요청에 응답하기 위해 HTML문서 형식을 제공해야 한다.
response로부터 얻은 출력스트림 객체인 out의 print 메소드를 이용해 23-28 라인처럼 HTML 코드를 직접 작성해야 한다.

2 컨텍스트 패스, Context Path

위에서 작성한 서블릿을 확인하기 위해 접속한 URL은 다음과 같다

```
http://localhost:8080/[myWeb] / helloServlet
```

http://localhost:8080 까지는 웹 서버를 지칭하며 뒤의 myWeb은 웹 서버에서 어떠한 서비스를 받을 것인지 지정하는 문자열이다. 웹 서버에서는 다양한 서비스를 제공할 수 있으며 이를 구분하기 위한 문자열을 컨텍스트 패스(Context Path)라고 부른다.

톰캣 서버에서 웹 어플리케이션의 서비스를 제공할 수 있게 해주려면 톰캣 서버에 웹 어플리케이션을 등록해야 하며 톰캣 서버의 server.xml 파일의 <Context> 태그를 사용한다. 이클립스를 사용하지 않을 때에는 일일이 개발자가 기술해야 하지만 이클립스는 프로젝트 단위로 자동 생성하게 된다.

톰캣 서버의 server.xml을 열어보면 이클립스에서 자동으로 추가해준 내용을 확인할 수 있다.

```
<Context docBase="myWeb" path="/myWeb"
         reloadable="true"
         source="org.eclipse.jst.jee.server:myWeb"/>
```

3 서블릿 클래스 구성

서블릿 클래스는 클라이언트를 통해 서비스된다는 사용되는 목적에 따라 그 형태가 정형화된 모습을 보인다. javax.servlet.http 패키지에서 제공하는 HttpServlet 클래스를 상속받아 구현해야 하며, 브라우저를 통해 외부에서 실행되도록 하기 위해 접근 제한자를 public으로 지정해야 한다.

```
public class [서블릿 클래스명] extends HttpServlet {
    // 접근제한자는 반드시 public
    // HttpServlet 클래스를 상속
}
```

3.1 GET 방식, POST 방식의 클라이언트 요청

클라이언트는 서버에 get과 post로 불리는 두 가지 방식 중 하나로 요청을 한다.

GET 방식은 서버에 있는 정보를 요청하여 가져오기 위해 설계되었다

URL에 쿼리 스트링을 추가하여 데이터를 전송하는 방식이며 형식은 다음과 같다

형식 : http://xxx.xxx.co.kr/servlet/login?id=tommy&name=arizona

240바이트까지 전달할 수 있으며 QUERYSTRING 환경변수를 통해 전달된다

URL노출로 보안성이 요구되는 경우에는 사용하면 안되며 검색엔진에서 검색단어 전송에 많이 이용한다

POST 방식은 서버로 정보를 올리기 위해 설계되었다

html header에 포함되어 전송되며 데이터크기의 제한은 없다

URL에 파라미터가 표시되지 않는다

* 쿼리 스트링

입력한 데이터를 서버로 전달하는 방식으로 '?'로 시작한다. 데이터는 '이름=값'의 형식을 취하며 여러 데이터들이 있을 경우 '&'로 구분한다.

3.2 doGet(), doPost()

서블릿은 클라이언트의 요청 방식에 따라 doGet() 혹은 doPost() 메소드가 호출된다

클라이언트의 요청 방식에 맞추어 HttpServlet 클래스의 doGet() 혹은 doPost() 메소드를 오버라이딩해야 한다.

- doGet() 메소드

```
public void doGet(HttpServletRequest req,  
                    HttpServletResponse resp  
                ) throws ServletException, IOException  
{  
  
}
```

req는 요청을 처리하기 위한 매개변수
resp는 응답을 처리하기 위한 매개변수

- doPost() 메소드

```
public void doPost(HttpServletRequest req,  
                    HttpServletResponse resp  
                    ) throws ServletException, IOException  
{  
  
}
```

req는 요청을 처리하기 위한 매개변수

resp는 응답을 처리하기 위한 매개변수

하나의 서블릿은 get과 post방식을 모두 처리할 수도 있으며 방식에 따라 다른 기능을 제공하는 경우가 많아 doGet() 메소드와 doPost() 메소드 둘 다 오버라이딩을 하기도 한다.

3.3 출력 스트림

서블릿의 결과로 HTML 문서를 제공하기 위해 출력 스트림을 얻어와야 한다.

출력 스트림을 얻어오기 전에 HTML 문서에 한글이 포함되어 있다면 한글이 깨지지 않도록 캐릭터 셋을 UTF-8로 지정해 주어야 한다.

```
response.setContentType("text/html;charset=UTF-8"); // UTF-8 지정  
PrintWriter out = response.getWriter(); // 출력 스트림 객체 얻기
```

4 서블릿 동작 원리

톰캣이 구동되면 자바가상머신(JVM)이 구동되어 자바 문법을 따르는 서블릿을 처리할 수 있는 환경이 제공된다. 이러한 환경을 서블릿 컨테이너라고 부른다. 서블릿의 실행은 서블릿 컨테이너에 의해 이루어진다.

4.1 서블릿 컨테이너

웹서버와 서블릿 사이의 통신을 지원

서블릿의 생명주기를 관리

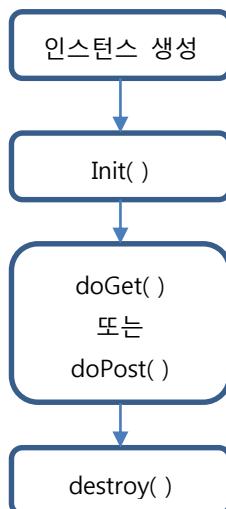
새로운 요청이 들어올 때마다 자바 스레드를 생성해서 사용자의 요청을 처리

컨테이너는 선언적인 방법으로 보안관련 내용을 설정할 수 있다



4.2 서블릿 라이프 사이클

서블릿은 클라이언트의 첫 번째 요청인 경우 서블릿 클래스를 찾아 메모리에 객체를 생성한다. 인스턴스화된 서블릿 객체는 메모리에 계속 남아 있게 되고 이후 서블릿이 호출되어도 서블릿 인스턴스가 다시 생성되지 않고 이미 로딩된 서블릿의 서비스를 받게된다.



첫 번째 요청일 경우, 객체가 생성되면서 `init()` 메소드를 한 번 호출하여 초기화 작업을 진행한다. 그 후 클라이언트의 요청이 있을 때마다 `doGet()` 또는 `doPost()` 메소드가 실행되며, 이후에는 초기화 작업을 하지 않는다. 또한 여러 클라이언트가 동시에 요청하더라도 스레드가 생성되며 처리가 되어 `doGet()`, `doPost()`의 수행 속도가 빠른 편이다.

최종적으로 서블릿이 서비스되지 않을 때 `destroy()` 메소드가 호출되는데 톰캣이 재시작되거나

종료되었을 때와 서블릿 내용이 변경되어 다시 컴파일했을 경우이다.

Chapter3. JSP 기본구조와 문법

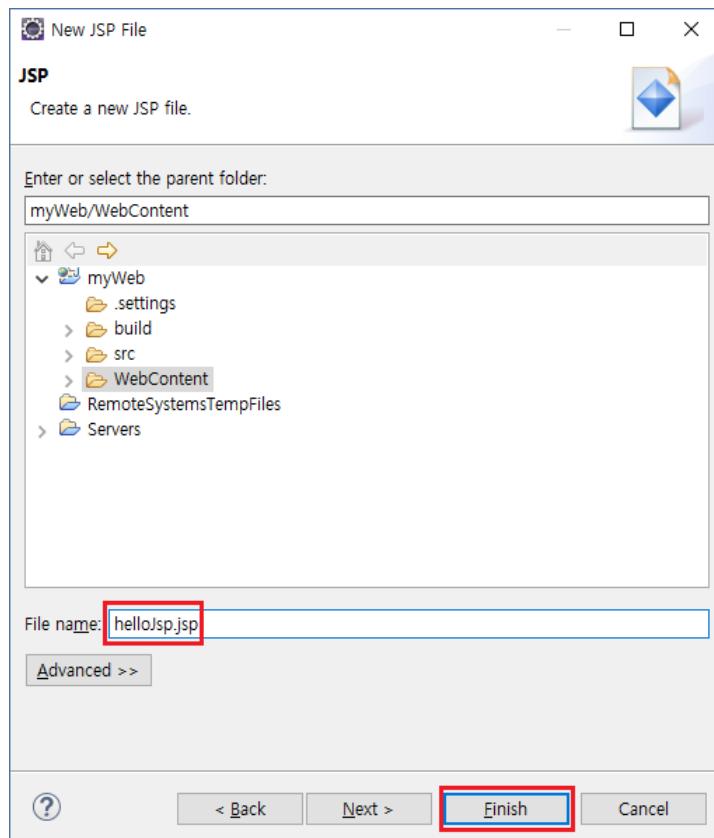
1 기본 JSP 프로그램 만들기

JSP를 이용해 기본 프로그램 만들기

- ① 만들어둔 프로젝트에 [File – New – JSP File] 선택

JSP File 이 안보인다면 Other 메뉴에서 jsp 검색

파일이름을 helloJsp.jsp 로 지정하며 생성



- ② 다음과 같이 작성

```
helloJsp.jsp
01 <%@ page language="java" contentType="text/html; charset=UTF-8"
02     pageEncoding="UTF-8"%>
03 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
04 "http://www.w3.org/TR/html4/loose.dtd">
05 <html>
```

```
06 <head>
07 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
08 <title>Hello JSP</title>
09 </head>
10 <body>
11   <h2>Hello Jsp</h2>
12 </body>
13 </html>
```

- ③ 브라우저를 이용해 <http://localhost:8080/myWeb/helloJsp.jsp> 으로 접속하면 결과 확인 가능



Hello Jsp

* JSP파일로 만들어진 페이지는 웹 서버 내부적으로 서블릿 코드를 생성한다

이클립스로 작업한 경우

(workspace 폴더) metadata plugins org.eclipse.wst.server.core tmp0 work Catalina localhost myWeb org WebContent jsp 내에 생성된 것을 확인할 수 있다.

2 JSP 기본 문법

JSP만의 문법에 대해 알아보도록 한다

2.1 주석

JSP에는 HTML, JSP, JAVA 코드 3가지가 섞여 있으므로 주석 또한 3가지를 적용할 수 있다

2.1.1 HTML 주석

웹 브라우저가 해석하지 않도록 하는 주석

```
<!-- 주석 -->
```

2.1.2 JSP 주석

JSP가 서블릿 코드를 변환될 때 포함되지 않도록 하는 주석

```
<%-- 주석--%>
```

2.1.3 Java 주석

JSP내의 자바 코딩 영역에서 사용되는 주석

```
// 한 줄 주석  
/* 문장 주석 */
```

2.2 JSP 기본 태그

종류	설명	형식
스크립트릿(scriptlet)	자바 코드를 기술	<% %>
선언(declaration)	변수와 메소드를 선언	<%! %>
표현식(expression)	계산식이나 함수 리턴을 문자열 형태로 출력	<%= %>

2.2.1 스크립트릿, Scriptlet

JSP에는 자바 코드와 HTML 코드가 섞여 있게 된다. HTML 태그는 정적인 페이지를 디자인하게 되고 자바 코드는 웹 서버에서 처리해야 할 문장들을 기술하게 된다. 이 때 사용되는 자바 언어를 서버 스크립트 언어라고 하며, JSP문서 내부에서 자바 코드를 사용하기 위해 제공되는 것이 스크립트릿 태그이다.

스크립트릿 태그 내에 정의된 내용은 JSP코드가 서블릿으로 변환될 때 `_jspService()`메소드 내부로 들어가게 된다.

- 스크립트릿 태그 기본 형식

```
<%
```

웹 서버에서 실행되어야하는 자바 코드

%>

2.2.2 선언문

JSP 페이지에서 변수나 메소드를 정의하기 위해 선언문을 사용한다

서블릿으로 변환될 때 스크립트릿과는 달리 클래스 영역에 정의된다

- 선언문 태그 기본 형식

```
<%!  
    변수 선언  
    메소드 정의  
%>
```

2.2.3 표현식

브라우저에 HTML 형태로 결과를 출력하기 위해서 간단한 방식으로 제공되는 태그

표현식으로 기술된 내용은 서블릿 컨테이너에서 스크립트 내의 `out.print()` 형식으로 변환된다

- 표현식 태그 기본 형식

```
<%=변수 %>  
<%=수식 %>  
<%=메소드호출 %>
```

2.3 JSP 지시자

JSP 페이지가 실행되면서 지시자에 설정된 내용을 JSP 페이지 전체에서 사용 가능하도록 한다

지시자에는 `page`, `include`, `taglib` 3가지가 있다

JSP 문서의 제일 위 부분에 작성한다

- 지시자 태그 기본 형식

```
<%@ 지시자 속성="값" ... %>
```

- 지시자 사용 목적

종류	목적
page	해양 JSP 페이지 전반적인 환경 설정
include	페이지에 다른 파일의 내용을 삽입
taglib	태그 라이브러리에서 기능을 사용할 수 있도록 제공

2.3.1 page 지시자

JSP 페이지에 여러 정보를 나타내기 위해 사용되는 JSP 문법

- language 속성

JSP에서 사용할 언어를 결정

```
<%@ page language="java" %>
```

JSP에서는 서버 스크립트 언어로 Java만 사용 가능하다. 따라서 language 속성으로 java만 기술할 수 있으며 디폴트로 java가 설정되어 있다.

- extends 속성

jsp페이지가 상속받을 상위 클래스를 기술

일반적으로 서블릿 컨테이너가 알아서 처리하며 개발자가 지정할 필요가 없다

```
<%@ page extends="javax.servlet.jsp.HttpJspBase" %>
```

- import 속성

java 소스와 마찬가지로 import할 클래스의 풀네임을 기술한다

JSP는 기본적으로 javax.servlet.*, javax.servlet.http.*, javax.servlet.jsp.* 3개의 패키지 내용을 import하고 있다

```
<%@ page extends="java.util.Calendar" %>
```

- session 속성

true / false 로 설정하며 세션을 사용할지 말지를 결정하는 속성

HttpSession 객체를 사용할 것인지 결정된다

기본적으로 true로 설정되어 있으며 세션을 사용하지 않으려면 false를 지정한다

```
<%@ page session="false" %>
```

- buffer 속성

JSP페이지의 출력 버퍼 크기를 설정, 디폴트는 8kb

none으로 설정될 경우 JspWriter 객체를 이용한 출력 시 버퍼를 사용하지 않는다

```
<%@ page buffer="8kb" %> <%-- 디폴트--%>
```

- autoFlush 속성

자동으로 버퍼를 비울지 설정

autoFlush를 false로 지정할 경우 버퍼가 가득 찰 경우 에러가 발생할 수 있다

autoFlush를 false로 지정하려면 buffer속성이 none이어서는 안 된다

```
<%@ page autoFlush="false" %>
```

autoFlush를 이용하지 않고 out 객체의 flush() 메소드를 이용할 수도 있다

- isThreadSafe 속성

프로세스를 스레드로 처리하여 수행속도와 효율성을 개선할 수 있지만 하나의 자원에 스레드들이 동시에 접근할 경우 무결성을 해칠 수 있게 된다

이런 상황을 대비하여 스레드들이 각각의 페이지에서 안전하게 동작(Thread Safe)할 수 있도록 설정한다

```
<%@ page isThreadSafe="false" %>
```

false로 지정할 경우 클라이언트의 요청에 한꺼번에 스레드를 만들어 처리하므로 객체들을 따로 동기화 시켜야한다

- info 속성

JSP 페이지에 대한 간략한 설명을 기록하는 용도

페이지에 대한 정보를 알려주는 역할을 한다

```
<%@ page info="페이지 정보" %>
```

- errorPage 속성

JSP 페이지에서 에러가 발생했을 때 보여줄 에러 페이지를 설정

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

- isErrorPage 속성

현재 페이지가 에러페이지인지 아닌지 설정하기 위한 속성

true로 설정되면 exception 객체를 사용하여 예외의 원인을 확인할 수 있게 된다

기본값은 false

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

- contentType 속성

JSP페이지의 MIME(Multipurpose Internet Mail Extensions) 타입을 결정한다

MIME는 서버가 클라이언트에게 보낼 내용에 대하여 미리 보내는 내용이며 헤더에 포함된다

웹 서버가 브라우저에 전송할 페이지가 html 일 경우 "text/html"을 지정

```
<%@ page contentType="text/html" %>
```

한글 HTML 문서로 전달할 때는 인코딩 방식 또한 UTF-8로 지정한다

```
<%@ page contentType="text/html;charset=UTF-8" %>
```

2.3.2 include 지시자

현재 페이지에 다른 HTML문서나 JSP 페이지의 내용을 삽입할 때 사용

include는 file이 유일한 속성

- file 속성

삽입할 파일의 URL을 기술

웹 사이트에서 공통적으로 보여주어야 할 내용이 있으면 include 지시자를 통해 표현한다

```
<%@ include file="파일명" %>
```

2.3.3 taglib 지시자

JSP에서 사용 가능한 태그 라이브러리를 지정

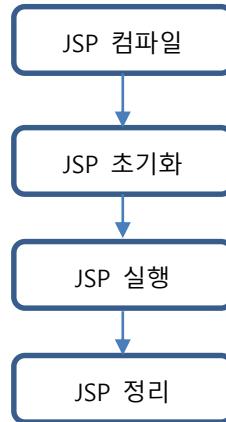
JSP 내의 불필요한 자바 코드를 줄일 수 있다

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
```

3 JSP 라이프 사이클

JSP의 라이프 사이클(생명 주기)는 서블릿과 유사

컴파일 - 초기화 - 실행 - 정리, 4단계를 거친다



3.1 JSP 컴파일

클라이언트(브라우저)에서 JSP를 요구할 경우 JSP엔진이 먼저 페이지를 컴파일 할 필요가 있는지 확인한다

JSP가 수정 된 일이 있다면 컴파일 수행

컴파일은 JSP구문분석, 서블릿으로 JSP 변환, 서블릿 컴파일 3단계를 거친다

3.2 JSP 초기화

컨테이너가 JSP 파일을 로드 한 후 서비스를 제공하기 전에 요청 `jspInit()` 메소드를 호출

3.3 JSP 실행

JSP 페이지 초기화가 완료되면 JSP 엔진은 `_jspService()` 메소드를 호출

3.4 JSP 정리

JSP 컨테이너에 의해 제거될 때 발생

`jspDestroy()`메소드가 수행되며 서블릿의 `destroy()`에 해당한다

Chapter4. JSP 내장 객체

1 내장 객체

JSP 내장 객체는 JSP 페이지에서 객체를 생성하는 과정 없이 바로 사용할 수 있는 객체들을 뜻 한다. JSP가 서블릿 파일로 변환될 때 서블릿 컨테이너가 자동으로 생성을 해주며 JSP 파일에서 사용할 때에는 생성되는 과정을 생각할 필요가 없다

test.jsp

```
01 <%@ page language="java" contentType="text/html; charset=UTF-8"
02     pageEncoding="UTF-8"%>
03 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
04 "http://www.w3.org/TR/html4/loose.dtd">
05 <html>
06 <head>
07 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
08 <title>Hello JSP</title>
09 </head>
10 <body>
11
12 <%
13     out.print("Hello JSP");
14 %>
15
16 </body>
17 </html>
```

test.jsp 소스의 13번째 줄처럼 out객체를 사용할 때 별도의 객체 생성과정 없이 사용가능한 것이 내장 객체이다. 반면, 서블릿에서 out객체를 사용하기 위해서는

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

와 같은 객체 생성이 필요하다

- 내장 객체의 용도에 따른 4가지 분류

내장 객체 분류	내장 객체
입출력 관련 객체	request response out
서블릿 관련 객체	page config
외부 환경 정보를 제공하는 객체	session application pageContext
예외 관련 객체	exception

2 out 내장 객체

서버에서 클라이언트로 열려있는 출력 스트림

JSP의 실행결과를 클라이언트의 브라우저로 출력할 때 효율적으로 사용할 수 있다

출력 전달 내장 객체

3 request 내장 객체

클라이언트의 요청과 관련된 정보가 저장되는 내장 객체

request 객체를 파악하면 클라이언트에서 서버로 전송된 데이터를 알 수 있다

- 클라이언트의 요청이 있을 때 관련된 정보들을 알 수 있는 메소드

반환타입	메소드 명	설명
String	getRemoteAddr()	웹 서버에 연결한 클라이언트의 IP주소를 구함
long	getContentLength()	클라이언트가 전송한 요청정보의 길이를 구함
String	getCharacterEncoding()	클라이언트가 요청 정보를 전송할 때 사용한 캐릭터셋을 구함
String	getContentType()	클라이언트가 요청 정보를 전송할 때 사용한 컨텐트 타입을 구함

String	getProtocol()	클라이언트가 요청한 프로토콜을 구함
String	getMethod()	데이터 전송 방식을 구함
String	getRequestURL()	요청 URL을 구함
String	getContextPath()	JSP페이지가 속한 웹어플리케이션의 컨텍스트패스 반환
String	getServerName()	연결할 때 사용한 서버이름을 구함
int	getServerPort()	사용중인 프로토콜을 구함

- request 객체의 요청 파라미터 관련 메소드

반환타입	메소드 명	설명
String	getParameter(String name)	지정한 이름의 파라미터가 있을 경우 첫 번째 파라미터의 값을 구함
String[]	getParameterValues(String name)	지정한 이름의 파라미터가 있을 경우 지정한 이름을 가진 파라미터의 모든 값을 String[]으로 구함
Enumeration	getParameterNames()	모든 파라미터 이름을 구함
Map	getParameterMap()	전송한 파라미터를 맵 형식으로 구함

4 response 내장 객체

클라이언트에 대한 응답을 처리하는 내장 객체

실행 결과를 브라우저로 되돌려 줄 때 사용

리다이렉트 기능을 많이 사용

* JSP 페이지 이동 방법 1. 리다이렉트

브라우저의 URL을 변경하도록 하여 페이지를 이동하는 방식

request와 response 객체가 유지되지 않는다

response 객체의 sendRedirect() 메소드를 사용하여 페이지 이동

* JSP 페이지 이동 방법 2. 포워딩

브라우저의 URL을 유지하며 문서의 내용(페이지)만 이동하는 방식

request 객체가 유지된다

requestDispatcher 객체로 접근한 후 forward() 메소드를 호출하여 페이지 이동

5 내장 객체의 영역(Scope)

내장 객체의 영역은 객체의 유효범위(유효기간)을 뜻한다

해당 객체가 얼마나 유지되는 지 지정해 주는 것

영역은 page, request, session, application으로 4가지로 분류된다

영역	설명
page	하나의 JSP페이지를 처리할 때 사용되는 영역
request	하나의 요청을 처리할 때 사용되는 영역
session	하나의 브라우저와 관련된 영역
application	하나의 웹 어플리케이션과 관련된 영역

5.1 page 영역

한 번의 클라이언트 요청에 하나의 JSP 페이지를 범위로 갖는다

클라이언트의 요청을 처리하는 JSP 페이지는 요청에 따라 새로운 page 영역을 갖게 되고 pageContext 내장 객체를 할당 받는다

해당 페이지 내에서만 사용할 수 있다

5.2 request 영역

클라이언트로부터 오는 한 번의 요청과 관련

클라이언트가 요청한 페이지와 요청을 받은 페이지 사이에서 request 내장 객체에 정보를 저장할 수 있다. 브라우저가 결과를 받으면 그 요청과 관련된 request 내장 객체는 사라진다. page 영역은 JSP 페이지만을 포함하며 request 영역은 하나의 요청을 처리하는데 사용되는 모든 JSP 페이지를 포함한다.

request 영역에 저장된 정보를 얻기 위해서는 request 객체가 필요하다

5.3 session 영역

웹 브라우저를 닫기 전까지 페이지를 이동하더라도 유지되며 웹 서버에서 제공하는 것
세션을 통해 웹 어플리케이션의 로그인 상태를 유지한다

5.4 application 영역

웹 어플리케이션에 속한 모든 페이지, 페이지에 대한 요청, 세션 모두 하나의 application 영역에 속한다. 모든 JSP 페이지는 하나의 application 내장 객체를 공유하고 있으며 이 application 내장 객체가 application 영역에 속한다.

Chapter5. JSP 액션 태그

1 액션 태그

클라이언트 혹은 서버에게 어떤 행동을 하도록 지시하는 태그를 뜻한다

액션 태그는 스크립트릿, 주석, 디렉티브와 함께 JSP 페이지를 이루는 요소 중 하나이다

태그 종류	설명
<jsp:forward>	다른 사이트로 이동할 때 사용 페이지의 흐름을 제어
<jsp:include>	정적 혹은 동적인 자원을 현재 페이지에 포함시킴 페이지를 모듈화할 때 사용
<jsp:param>	<jsp:forward>, <jsp:include>, <jsp:plugin>과 같이 사용됨 파라미터를 추가할 때 사용
<jsp:useBean>	빈(Bean)을 생성하고 사용하기 위한 환경을 정의
<jsp:setProperty>	빈에서 속성 값을 할당
<jsp:getProperty>	빈에서 속성 값을 얻어올 때 사용

2 <jsp:param> 액션 태그

파라미터 값을 전달할 때 사용

단독으로 사용하지 않고 <jsp:include>나 <jsp:forward> 태그의 내부에 기술하여 사용

- 형식

```
<jsp:param name="파라미터 변수" value="값"/>
```

3 <jsp:forward> 액션 태그

페이지를 이동할 때 사용

포워드 방식으로 페이지를 이동

기존 request 연결을 유지하면서 서버상의 url로 request 정보를 전달

- 형식

```
<jsp:forward page="이동할페이지">  
<jsp:param name="변수명" value="값" />  
</jsp:forward>
```

4 <jsp:include> 액션 태그

모든 페이지에서 공통적으로 사용되어야 하는 내용에 대해서 코드를 복사 붙여넣기 하는 것보다 각각 JSP 페이지에 구성을 하고 액션태그를 이용해 페이지를 포함시키는 것이 좋다
공통적으로 적용된 부분을 한꺼번에 수정할 수 있게 되어 유지보수가 수월해진다

- 형식1

```
<jsp:forward page="포함할 페이지 이름" flush="true" 또는 "false" />
```

- 형식2

```
<jsp:forward page="포함할 페이지 이름" flush="true">  
<jsp:param name="변수명" value="값" />  
</jsp:forward>
```

5 <jsp:plugin> 액션 태그

자바 애플리케이션 또는 자바빈즈 컴포넌트를 클라이언트로 다운받아 사용할 수 있도록 브라우저에 맞는 HTML 코드를 생성해주는 역할을 한다. 서버 측에서 사용되는 컴포넌트의 경우 서버에 부하를 주게되는데 plugin 액션 태그를 이용해 서버의 부하를 줄여줄 수 있다

- 형식

```
<jsp:plugin type="플러그인 타입" codebase="클래스 파일 위치"  
code="불리울 클래스 파일" width="가로" height="세로">  
<jsp:params>  
<jsp:param name="파라미터이름" value="파라미터값"/>  
<jsp:params>  
</jsp:plugin>
```

- plugin 태그의 속성

속성	사용법	설명
type	type="appletbean"	타입 설정
code	code="code.class"	사용할 클래스 파일 설정
width	width="800"	플러그인 표시 영역 너비 지정
height	height="600"	플러그인 표시 영역 높이 지정
codebase	codebase="/"	플러그인 파일 위치 지정
name	name="pluginOne"	플러그인 식별을 위한 이름 지정
align	align="left"	정렬방식 지정
vspace	vspace="10"	수직여백 설정
hspace	hspace="10"	수평여백 설정
jreversion	jreversion="1.8"	플러그인을 실행할 JRE 버전 입력
iepluginurl	iepluginurl="url"	IE전용 플러그인 다운 URL 입력
nspluginurl	nspluginurl="url"	넷스케이프 전용 플러그인 다운 URL 입력

6 <jsp:useBean> 액션 태그

객체의 이름과 사용범위, 빈의 저장위치 등을 통해서 객체가 생성된다. JSP의 자바 코드에서는 action의 id 특성에 지정된 값을 통해서 객체를 참조한다.

- 형식

```
<jsp:useBean id="빈 이름" scope="범위" class="빈의 저장위치"/>
```

id : 객체 인스턴스를 식별하는 이름

scope : 객체 참조의 유효범위(default : page)

class : 완전한 형태의 클래스 이름

7 <jsp:setProperty> 액션 태그

빈의 속성에 값을 설정하는 태그

- 형식

```
<jsp:setProperty name = "빈 이름" property="프로퍼티 명" value="저장할 값"/>
```

name : <jsp:useBean> 태그에 정의된 빈 인스턴스 이름

property : 값을 설정하고자 하는 빈 속성의 이름 , 설정 시 servletRequest안의 모든 인자들 중 빈 속성과 데이터 형이 일치하는 것을 찾아 각각의 속성들을 각각의 인자 값으로 설정한다.

value : 빈 속성에 설정할 값

8 <jsp:getProperty> 액션 태그

빈의 속성값을 얻는데 사용

- 형식

```
<jsp:getProperty name = "빈 이름" property="프로퍼티 명"/>
```

name : 속성을 얻고자 하는 빈 인스턴스의 이름

property : 얻고자 하는 속성의 이름

Chapter6. 자바 빈, Java Bean

1 자바 빈

프로그램에서 사용될 여러 개의 정보를 모아놓는 데이터 저장소의 역할을 하는 클래스

정보 은닉(Data Hiding) 개념이 적용된다

자바의 클래스이며 필드와 getter/setter 메소드를 하나의 쌍으로 갖는 특별한 클래스를 뜻한다

JSP에서 <jsp:useBean>, <jsp:setProperty>, <jsp:getProperty> 액션 태그를 사용한다

2 자바 빈 설계 규약

JavaBean은 특별한 목적이 있는 클래스지만 일반적인 클래스의 형태를 가지고 있기 때문에 사용하다 보면 단순히 일반 클래스가 되는 경향이 있다. 이를 방지하기 위해 JavaBean만의 설계규약이 존재한다.

- 멤버 변수마다 별도의 getter/setter 메소드가 존재해야 한다
- getter 메소드는 파라미터가 존재하지 않는다
- setter 메소드는 파라미터가 반드시 하나 이상 존재한다
- 반드시 패키지화 한다
- 멤버 변수는 private 형으로 지정하여 선언한다(정보 은닉)

3 자바 빈을 이용한 간단한 웹 어플리케이션

WebContent/sample/simpleForm.jsp

```
01 <%@ page language="java" contentType="text/html; charset=UTF-8"
02     pageEncoding="UTF-8"%>
03 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
04 "http://www.w3.org/TR/html4/loose.dtd">
05 <html>
06 <head>
07 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
08 <title>Insert title here</title>
09 </head>
10 <body>
11     <h1>간단한 자바빈즈 프로그램 </h1>
```

```

12      <hr color="red"/>
13      <form method="post" action="simpleBean.jsp">
14          메세지 : <input type="text" name="message" />
15          <input type="submit" value="전송">
16      </form>
17  </body>
18 </html>

```

src/sample/SimpleData.jsp

```

01 package sample;
02
03 public class SimpleData{
04     private String message;
05
06     public void setMessage(String message){
07         this.message = message;
08     }
09
10    public String getMessage(){
11        return message;
12    }
13 }

```

WebContent/sample/simpleBean.jsp

```

01 <%@ page language="java" contentType="text/html; charset=UTF-8"
02             pageEncoding="UTF-8"%>
03 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
04 "http://www.w3.org/TR/html4/loose.dtd">
05
06 <%
07 request.setCharacterEncoding("utf-8");
08 //String message = request.getParameter("message");
09 %>
10

```

```
11 <jsp:useBean id="msg" class="sample.SimpleData" />
12 <%-- SimpleData msg = new SimpleDataO; --%>
13
14 <jsp:setProperty name="msg" property="message" />
15 <%-- msg.setMessage (?); --%>
16
17 <html>
18 <head>
19 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
20 <title>Insert title here</title>
21 </head>
22 <body>
23     <h1>간단한 빈즈 프로그램 결과</h1>
24     <hr color="red"></hr><br>
25     <font size="5">
26         메세지 : <jsp:getProperty name="msg" property="message" />
27         <%-- <%=message%> --%>
28     </font>
29 </body>
30 </html>
```

간단한 자바빈즈 프로그램

메세지 :

간단한 빈즈 프로그램 결과

메세지 : Hello

Chapter7. 표현언어와 JSTL

1 표현언어, Expression Language

값을 웹페이지에 표현 언어는 값을 표현하는데 사용되는 새로운 스크립트 언어로서 JSP2.0 규약에서 새롭게 추가된 기능

JSP에서 출력을 위해서 표현식을 사용

- 표현 언어에서 제공하는 기능

- ① JSP의 네 가지 기본 객체가 제공하는 영역의 속성 사용
- ② 집합 객체에 대한 접근 방법 제공
- ③ 수치연산, 관계연산, 논리연산자 제공
- ④ 자바 클래스 메소드 호출 기능 제공
- ⑤ 표현 언어만의 기본 객체 제공

JSP에서는 스크립트 요소를 제공하며 그 중 표현식 스크립트는 계산식, 함수 호출 결과를 문자열 형태로 출력해주는 역할을 한다. 표현언어는 표현식보다 사용 방법이 간단하고 문법 체계가 직관적이어서 더욱 쉽게 사용 가능하다.

- 표현식

```
<%=표현식%>
```

- 표현 언어

```
${표현언어}
```

- "Hello EL"을 세가지 방법으로 출력한 예제

expr.jsp

```
01 <%@ page language="java" contentType="text/html; charset=UTF-8"
02     pageEncoding="UTF-8"%>
03 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
04 "http://www.w3.org/TR/html4/loose.dtd">
05 <html>
06 <head>
07 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
08 <title>Insert title here</title>
```

```

09 </head>
10 <body>
11
12 ${"Hello EL" } <br> <!-- 표현 언어 -->
13 <%="Hello EL" %> <br> <!-- 표현식 -->
14 <% out.println("Hello EL"); %> <br> <!-- 스크립트릿 -->
15
16 </body>
17 </html>

```

1.1 표현 언어에서 사용되는 자료형

표현 언어에서는 정수형, 실수형, 논리형, null 등의 데이터 타입을 사용할 수 있다

```

exprDatatype.jsp

```

```

01 <%@ page language="java" contentType="text/html; charset=UTF-8"
02     pageEncoding="UTF-8"%>
03 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
04 "http://www.w3.org/TR/html4/loose.dtd">
05 <html>
06 <head>
07 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
08 <title>Insert title here</title>
09 </head>
10 <body>
11
12 정수형 : ${15 } <br>
13 실수형 : ${6.7 } <br>
14 문자열 : ${"안녕하세요" } <br>
15 논리형 : ${false } <br>
16 null : ${null } <br>
17
18 </body>
19 </html>

```

1.2 표현 언어에서 사용되는 연산자

표현 언어로 접근된 데이터를 조작 및 비교, 연산을 할 수 있도록 연산자의 사용이 가능하다

종류	연산자
산술	+ - * /(or div) %(or mod)
관계	==(or eq) !=(or ne) <(or lt) >(or gt) <=(or le) >=(or ge)
조건	a ? b : c
논리	&&(or and) (or or) !(or not)
null 검사	empty

2 표현 언어의 기본객체

EL에서는 11개의 기본객체를 제공하고 있다

기본 객체	설명
pageContext	JSP의 page 기본 객체와 동일
pageScope	pageCntext 기본 객체에 저장된 속성의 <속성, 값> 매핑을 지정한 Map 객체
requestScope	request 기본 객체에 저장된 속성의 <속성, 값> 매핑을 지정한 Map 객체
sessionScope	session 기본 객체에 저장된 속성의 <속성, 값> 매핑을 지정한 Map 객체
applicationScope	application 기본 객체에 저장된 속성의 <속성, 값> 매핑을 지정한 Map 객체
param	요청 파라미터의 <파라미터이름, 값> 매핑을 저장한 Map 객체 파라미터 값의 타입은 String으로서, request.Parameter(이름)의 결과와 동일
paramValues	요청 파라미터의 <파라미터이름, 값> 매핑을 저장한 Map 객체 파라미터 값의 타입은 String으로서, request.ParameterValues(이름)의 결과와 동일
header	요청 정보의 <헤더이름, 값> 매핑을 지정한 Map 객체 request.getHeader(이름)의 결과와 동일

headerValues	요청 정보의 <헤더이름, 값> 매핑을 지정한 Map객체 request.getHeaders(이름)의 결과와 동일
cookie	<쿠키이름, Cookie> 매핑을 저장한 Map객체 request.getCookie()로 구한 Cookie 배열로부터 매핑을 생성
initParam	초기화 파라미터의 <이름, 값> 매핑을 저장한 Map객체 application.getInitParameter()결과와 동일

2.1 표현언어로 로그인 폼 정보 활용하기

login.jsp

```

01 <%@ page language="java" contentType="text/html; charset=UTF-8"
02     pageEncoding="UTF-8"%>
03 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
04 "http://www.w3.org/TR/html4/loose.dtd">
05 <html>
06 <head>
07 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
08 <title>EL</title>
09 </head>
10 <body>
11
12 <form method="get" action="loginEx.jsp">
13     <label for="id">아이디 : </label> <input type="text" name="id" id="id" /><br>
14
15     <label for="pass"> 암호 : </label> <input type="password" name="pass"
16             id="pass" /><br> <input type="submit" value="로그인" />
17 </form>
18
19 </body>
20 </html>
```

loginEx.jsp

```

01 <%@ page language="java" contentType="text/html; charset=UTF-8"
02     pageEncoding="UTF-8"%>
```

```
03 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
04 "http://www.w3.org/TR/html4/loose.dtd">
05 <html>
06 <head>
07 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
08 <title>EL</title>
09 </head>
10 <body>
11
12 <h1>표현식 버전</h1><hr>
13 아이디 : <%=request.getParameter("id") %><br>
14 암호 : <%=request.getParameter("pass") %><br><br>
15
16 <h1>EL 버전</h1><hr>
17 아이디 : ${param.id }<br>
18 암호 : ${param.pass }<br><br>
19
20 </body>
21 </html>
```

아이디 : admin
암호 :
로그인

표현식 버전

아이디 : admin

암호 : 1234

EL 버전

아이디 : admin

암호 : 1234

3 표현 언어에서 내장 객체 사용하기

웹 어플리케이션을 JSP로 구현하는데 필요한 정보를 JSP 내장 객체에 속성 값으로 저장하여 사용한다. 속성에 저장된 값을 표현 언어에서도 접근 가능하다.

범위	내장 객체	설명
page	pageScope	page 객체에 저장된 속성의 <속성, 값>매핑을 저장한 Map 객체
request	requestScope	request 객체에 저장된 속성의 <속성, 값>매핑을 저장한 Map 객체
session	sessionScope	session 객체에 저장된 속성의 <속성, 값>매핑을 저장한 Map 객체
application	application	application 객체에 저장된 속성의 <속성, 값>매핑을 저장한 Map 객체

- JSP 내장 객체의 정보를 주고 받기 위한 메소드

반환타입	메소드	설명
void	setAttribute(String name, Object value)	name에 값을 설정
Object	getAttribute(String name)	name에 설정된 값을 얻음
Object	getAttributeNames()	현재 객체에 관련된 모든 속성 이름을 얻어냄
void	removeAttribute(String name)	name에 설정된 값을 제거

4 JSTL, JSP Standard Tag Library

JSP에서 사용 가능한 표준 태그 라이브러리

JSTL 태그를 사용하면 JSP 코드가 깔끔해지고 가독성이 좋아진다

JSP 페이지에서 논리적인 판단, 반복문의 처리, 데이터베이스 처리 등의 처리를 하는 코드를 깔끔하게 작성하기 위해서 커스텀 태그를 많이 작성하게 되는데 이런 중복된 노력을 제거하기 위해서 나온 것

4.1 JSTL에서 제공해주는 기능

- 간단한 프로그램 로직 구현(자바의 변수 선언, if문, for문 등)
- 다른 JSP 페이지 호출
- 날짜, 시간, 숫자의 포맷
- 데이터베이스로의 입력, 수정, 삭제, 조회(CRUD)
- XML 문서의 처리
- 문자열 처리 함수 호출

4.2 JSTL에서 제공하는 5가지 커스텀 태그

커스텀 태그	설명
기본 기능(core)	변수 선언, 실행 흐름 제어, 다른 JSP 페이지로 제어 이동
형식화(format)	숫자, 날짜, 시간의 포맷을 지정 국제화, 다국어 지원 기능
데이터베이스(sql)	데이터베이스 CRUD 지원
XML 처리(xml)	XML 문서를 처리할 때 필요한 기능 지원
함수 처리(functions)	문자열 처리 함수 제공

4.3 JSTL taglib 지시자

JSTL이 제공해 주는 기능을 사용하기 위해서는 taglib 지시자를 추가해주어야 한다

uri로 사용하려는 라이브러리를 지정하고 prefix 속성으로 접두사를 지정한다

- 형식

```
<%@ taglib uri="라이브러리 식별자" prefix="접두사" %>
```

- 태그 라이브러리의 URI 식별자와 접두사

기능	prefix	기본 URI
기본 기능	c	http://java.sun.com/jsp/jstl/core
형식화	fmt	http://java.sun.com/jsp/fmt
데이터베이스 작업	sql	http://java.sun.com/jsp/sql
XML 처리	x	http://java.sun.com/jsp/xml
함수 처리	fn	http://java.sun.com/jsp/jstl/fn

4.4 JSTL Core 태그

5종류의 라이브러리들 중에서 가장 자주 사용되는 태그

태그	설명
<c:set>	변수 값 설정
<c:remove>	변수 값 제거
<c:if>	조건문
<c:choose>	여러 조건을 처리할 때
<c:forEach>	반복 처리
<c:forTokens>	구분자로 분리된 각각의 토큰 처리
<c:import>	외부 자원을 url을 지정하여 가져와 사용
<c:redirect>	지정한 경로로 이동
<c:url>	url 재 작성
<c:out>	데이터 출력
<c:catch>	예외 처리

4.4.1 <c:set>과 <c:remove> 태그

<cset> 태그는 해당 범위 내에 속성을 생성하고 속성 값을 지정하는 데 사용

- <cset> 태그 형식

```
<cset var="변수명" value="저장할값" [scope="{page | request | session | application}"]>
```

var : 변수 이름을 String 형으로 지정

value : 변수에 저장할 값을 지정

scope : 변수의 효력이 발휘되는 영역 지정. 기본 값은 page

<c:remove> 태그는 해당 scope에 있는 변수를 제거하는 데 사용

- <c:remove> 태그 형식

```
<c:remove var="변수명" [scope="{page | request | session | application}"]>
```

var와 scope가 <c:set> 했을 때와 같아야 변수 제거 가능

4.4.2 흐름 제어 태그

<c:if> 태그는 자바의 if문과 비슷한 기능

else 문이 없는 단순 if문의 구조와 같다

- <c:if> 태그 형식

```
<c:if test="조건식">
```

참일 경우 수행

```
</c:if>
```

<c:choose> 태그는 자바의 if - else문과 비슷한 기능

순서대로 test 조건을 검사하며 참일 경우 해당 몸체 수행

- <c:choose> 태그 형식

```
<c:choose>
```

```
  <c:when test="조건1"> 몸체1 </c:when>
```

```
  <c:when test="조건2"> 몸체2 </c:when>
```

```
  <c:when test="조건3"> 몸체3 </c:when>
```

```
</c:choose>
```

<c:forEach> 태그는 반복 수행을 할 때 사용

배열이나 컬렉션, 맵 등과 같은 데이터 집합체에 저장되어 있는 값들을 순차적으로 처리할 때 사용하는 태그

- <c:forEach> 태그 형식1

```
<c:forEach [var="변수명" items="집합체"]>
```

```
    몸체
```

```
</c:forEach>
```

또한, 인덱스를 사용하여 원하는 구간만큼 반복횟수를 지정할 수 있다

- <c:forEach> 태그 형식2

```
<c:forEach [var="변수명"] begin="시작값" end="끝값" [step="증가치"]>  
    몸체  
</c:forEach>
```

<c:forTokens> 태그는 문자열을 구분자로 분리해서 하나씩 추출할 때 사용

- <c:forTokens> 태그 형식

```
<c:forTokens var="토큰을 저장할 변수" items="토큰으로 나눈 문자열" delims="구분자">  
    몸체  
</c:forTokens>
```

4.4.3 <c:import>

다른 페이지의 내용을 포함시키기 위해 사용

페이지의 내용을 변수에 저장할 수 있음

- <c:import> 태그 형식

```
<c:import url="URL" [var="변수명"] [scope="영역"] [charEncoding="charEncoding"]>  
</c:import>
```

4.4.4 <curl>

이후에 여러 번 반복되어 사용될 수 있는 주소를 변수에 저장하기 위해

- <curl> 태그 형식

```
<curl value="URL" [var="변수명"] [scope="영역"]>  
</curl>
```

4.4.5 <c:redirect>

지정한 페이지로 이동시키기 위해 사용

- <c:redirect> 태그 형식

```
<c:redirect url="URL" [context="경로명"]>
```

4.4.6 <c:out>

value 속성에 지정한 문자열 혹은 변수의 내용을 출력할 때 사용하는 태그

- <c:out> 태그 형식

```
<c:out value="value" [default="기본값"]>
```

지정한 출력값이 없을 경우 출력될 기본값을 지정할 수 있다

4.4.7 <c:catch>

예외처리를 위한 태그

- <c:catch> 태그 형식

```
<c:catch var="변수 이름" >
```

예외가 발생할 수 있는 코드

```
</c:catch>
```

var : 예외의 내용이 들어가는 변수

4.5 JSTL fmt

JSTL을 국제화 하기 위한 목적으로 JSP 페이지에서 다양한 언어를 지원 받을 수 있도록 한다
또한, 날짜와 숫자의 형식을 다루는 데 사용된다

기능	태그	설명
숫자 날짜 형식	formatNumber	포맷에 맞는 숫자를 출력
	formatDate	날짜 정보를 포맷팅하여 출력
	parseDate	문자열을 날짜 형식으로 파싱
	parseNumber	문자열을 숫자 형식으로 파싱

	setTimeZone	시간대를 지정
	timeZone	시간대를 지정
로케일 지정	setLocale	국제화 태그들의 로케일을 지정
	requestEncoding	요청 파라미터의 인코딩 지정
메시지 처리	bundle	태그 몸체에서 사용하는 리소스 번들 지정
	message(param)	메시지 출력
	setBundle	특정 리소스 번들을 사용할 수 있도록 로딩

- JSTL fmt 태그를 사용하기 위한 taglib 지시자

```
<${@ taglib prefix="fmt" uri="http://java.sun.com/jstl/fmt"%>
```

4.5.1 숫자 날짜 형식 지정 태그 1 <fmt:formatNumber>

숫자 형식을 지정하기 위한 태그로 사용

- <fmt:formatNumber> 태그 형식

```
<fmt:formatNumber value="수치 데이터"
    [type="{number | currency | percent}"]
    [pattern="패턴"]
    [currencySymbol="화폐 단위"]
    [groupingUsed="{true | false}"]
    [var="변수 이름"]
    [scope="page | request | session | application"] />
```

value : 형식화할 수치 데이터

type : 숫자, 통화 퍼센트 중 표시될 형식 지정

pattern : 지정한 형식 패턴

currencySymbol : 통화 기호, 통화 형식일 때만 적용

groupingUsed : 콤마처럼 단위를 구분할 때 사용하는 기호의 표시 여부 결정

var : 형식 출력 결과 문자열을 담는 scope에 해당하는 변수를 지정

scope : var 속성에 지정한 변수가 효력을 발휘하는 영역 지정

구분 기호를 표시할 지 결정하는 groupingUsed가 기본적으로 true로 지정되어 있어 value 속성에 지정한 수치 데이터를 출력할 때 단위를 구분하기 위해 콤마를 세자리 마다 한 번씩 찍게된다.

표시하지 않으려면 false로 지정한다.

type 속성에 percent를 지정하면 수치 데이터를 퍼센트로 표시하게 된다. 이 때 0.5를 데이터로 사용하면 50%로 출력하므로 수치 지정에 주의를 기울여야 한다.

통화 형식으로 출력하기 위해서는 type에서 currency를 지정한다.

4.5.2 숫자 날짜 형식 지정 태그 2 <fmt:formatDate>

날짜 형식을 지정하기 위한 태그로 사용

- <fmt:formatDate> 태그 형식

```
<fmt:formatDate value="date"  
[type="{time | date | both}"]  
[dateStyle="{default | short | medium | long | full}"]  
[timeStyle="{default | short | medium | long | full}"]  
[pattern="customPattern"]  
[TimeZone="TimeZone"]  
[var="변수 이름"]  
[scope="page | request | session | application"] />
```

value : 형식화될 Date와 time (java.util.Date 타입 사용)

type : 시간, 날짜, 시간날짜 모두 중 형식화할 데이터의 형태를 지정

dateStyle : 미리 정의된 날짜 형식 중 선택

timeStyle : 미리 정의된 시간 형식 중 선택

pattern : 사용자가 지정한 형식을 사용

TimeZone : 형식화 시간에 나타날 타임존

var : 형식 출력 결과 문자열을 담는 scope에 해당하는 변수를 지정

scope : var 속성에 지정한 변수가 효력을 발휘하는 영역 지정

날짜만 출력하고 싶다면 type에 date를 지정하고, 시간만 출력하고 싶다면 time을 지정한다. 둘 모두를 출력하고자 한다면 both를 지정하면 된다. 기본적으로 제공하는 형식을 사용하지 않고 사용자가 직접 형식을 바꾸고 싶다면 pattern에 yy, MM, dd, hh, mm, ss 등의 날짜 시간 형식 지정자를 사용하여 표현할 수 있다.

Chapter8. JDBC

1 JDBC, Java Database Connectivity

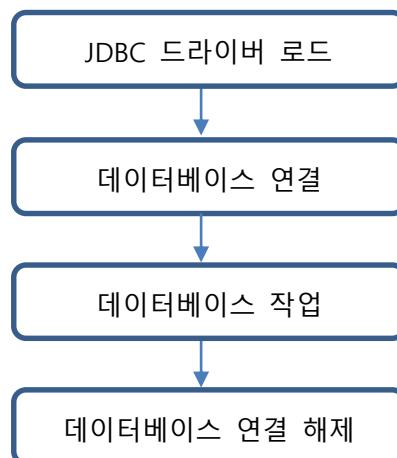
자바에서는 JDBC라는 강력한 도구를 제공해 자바 프로그램에서 데이터베이스를 처리할 수 있게 해준다. JDBC는 일관성 있는 방식으로 데이터베이스에 접근할 수 있는 API를 제공하는 클래스들이라고 볼 수 있다.

데이터를의 집합이라고 볼 수 있는 데이터베이스는 데이터베이스 관리 시스템(DBMS, Database Management System)에 의해 관리된다. DBMS에는 오라클, MSSQL Server, MySql 등이 있지만 자바 웹 프로젝트에서는 오라클 DBMS를 주로 사용하게 된다.

자바와 데이터베이스는 별도의 시스템으로 동작하기 때문에 자바에서 제공되는 순수 API에는 데이터베이스에 접근할 수 있는 기능을 가진 클래스가 제공되지 않는다. 자바 프로그램에서 데이터베이스 작업이 필요할 때 사용할 수 있도록 제공되는 것이 JDBC이다.

2 JDBC 동작

JDBC를 이용한 데이터베이스와 연결에는 4단계를 거쳐 동작하게 된다.



2.1 JDBC 드라이버 로드

자바에서 데이터베이스 연결을 위한 JDBC 드라이버는 데이터베이스 제작 업체에서 제공한다. 오라클이 설치된 폴더에서 JDBC 드라이버를 얻어와 자바 개발 환경에 설치해야만 드라이버를 사용할 수 있다.

JDBC 드라이버의 이름은 ojdbcX.jar 형식이며 버전별로 X가 달라질 수 있다.

- 오라클 11g 익스프레스 에디션을 설치할 경우 경로 예시

```
C:\oradlexe\app\oracle\product\11.2.0\server\jdbc\lib
```

JDBC 드라이버 파일(.jar)을 복사하여 Dynamic Web Project의 WEB-INF 폴더 내의 lib 폴더에 복사 넣기하면 드라이버가 설치된다.

우선 JDBC의 기능을 사용할 수 있도록 JDBC 클래스를 import 해 주어야 한다. JDBC 클래스는 java.sql 패키지에 존재한다.

- JDBC 클래스, 인터페이스 import

```
import java.sql.*;
```

이후 JDBC 클래스를 로드한다.

```
Class.forName("oracle.jdbc.driver.OracleDriver");
```

java.lang.Class 클래스의 forName 메소드는 문자열 형태로 전달된 클래스를 JVM으로 로드해주는 기능을 가지고 있다. Oracle JDBC 드라이버를 지정해 로드하여 DriverManager에 등록이 되도록 하는 코드이다.

2.2 데이터베이스 연결

JDBC 드라이버를 로딩한 후 데이터 베이스에 연결하기 위해 Connection 객체를 사용할 수 있다. Connection 객체는 데이터베이스에 연결해 작업을 수행할 수 있도록 도와주는 객체이다. DriverManager 클래스의 getConnection() 메소드를 호출하여 사용할 수 있다.

- Connection 객체 얻기

```
Connection conn = DriverManager.getConnection(url, uid, pwd);
```

url : JDBC 형식 URL

uid : 사용자 ID

pwd : 패스워드

Connection 객체를 얻기 위해 URL, UID, PWD를 지정해야 하는데 URL은 JDBC 형식의 URL이고 UID와 PWD는 사용자명과 패스워드를 뜻한다.

- JDBC 형식으로 오라클 DMBS를 지정하는 형식 예

```
jdbcoracle:thin:[hostname][:port]:dbname
```

익스프레스 에디션을 설치하였다면 dbname은 XE 이다.

- 오라클 익스프레스 에디션의 scott 계정에 Connection 객체 얻기

```
Connection conn = DriverManager.getConnection(  
    "jdbcoracle:thin:@localhost:1521:XE",  
    "scott",  
    "tiger");
```

이후 데이터베이스 작업을 진행할 수 있다.

데이터베이스 작업은 conn 객체를 통해 데이터베이스 작업을 할 수 있으며 모든 데이터 베이스 작업이 끝나면 연결을 종료시켜주면 된다.

- 데이터 베이스 연결 끊기

```
conn.close();
```

2.3 데이터베이스 작업

데이터베이스 연결을 한 후 실제 SQL문을 수행하기 위해서 Statement 객체를 생성한다.

데이터베이스 연결에 사용한 conn 객체를 통해 createStatement() 메소드를 호출해 Statement 객체를 얻어올 수 있다

- Statement 객체 얻기

```
Statement stmt = conn.createStatement();
```

stmt : 표준 SQL문을 수행하기 위한 Statement 객체

데이터베이스 작업이 완료되면 Statement 객체를 반환할 필요가 있어 close()메소드를 호출해야 한다.

- Statement 객체 닫기

```
stmt.close();
```

Statement 객체를 사용해 SQL 표준 쿼리문을 자바 프로그램에서 사용할 수 있다. 쿼리를 실행시키는 메소드는 두가지가 있다

메소드	설명
executeQuery	select 문과 같이 결과값이 여러 개의 레코드로 구해지는 경우 사용
executeUpdate	insert, update, delete 문과 같은 내부적으로 테이블 변경이 있고 결과 값이 없는 경우 사용

executeQuery 메소드의 결과는 ResultSet으로 받게 된다

- Select 구문 예시

```
String sql = "Select * from emp";
ResultSet rs = stmt.executeQuery(sql);
```

executeQuery 메소드의 리턴인 ResultSet 객체는 여러 개의 행으로 결과값을 받을 수 있으며 각 행에 접근할 수 있도록 Iterator와 비슷한 메소드를 제공한다.

- ResultSet의 대표 메소드

메소드	설명
next()	현재 행에서 한 행 앞으로 이동
previous()	현재 행에서 한 행 뒤로 이동
first()	첫 번째 행으로 이동
last()	마지막 행으로 이동

3 JSP에서 Database 연동하기

- 예제 테이블

```
CREATE TABLE "MEMBER" (
    "ID" VARCHAR2(20) NOT NULL,
    "PASSWD" VARCHAR2(20),
    "NAME" VARCHAR2(20),
    "MEM_NUM1" VARCHAR2(6),
```

```

"MEM_NUM2" VARCHAR2(7),
"E.MAIL" VARCHAR2(30),
"PHONE" VARCHAR2(30),
"ZIPCODE" VARCHAR2(7),
"ADDRESS" VARCHAR2(60),
"JOB" VARCHAR2(30),
PRIMARY KEY ("ID")
);

```

- 예제 CSS

WebContent/member/style.css

```

01 @CHARSET "EUC-KR"
02 BODY { FONT-SIZE : 9pt; COLOR : black; LINE-HEIGHT : 160%; }
03 TD { FONT-SIZE : 9pt; COLOR : black; LINE-HEIGHT : 160%; }
04 SELECT { FONT-SIZE : 9pt; COLOR : black; LINE-HEIGHT : 160%; }
05 DIV { FONT-SIZE : 9pt; COLOR : black; LINE-HEIGHT : 160%; }
06 FORM { FONT-SIZE : 9pt; COLOR : black; LINE-HEIGHT : 160%; }
07 TEXTAREA {
    BORDER-RIGHT : 1px solid #999999; BORDER-TOP : 1px solid #999999;
    FONT-SIZE : 9pt; BORDER-LEFT : 1px solid #999999; COLOR : BLACK;
    BORDER-BOTTOM : 1px solid #999999; BACKGROUND-COLOR : white;
}
08 INPUT {
    BORDER-RIGHT : 1px solid #999999; BORDER-TOP : 1px solid
    FONT-SIZE : 9pt; BORDER-LEFT : 1px solid #999999; COLOR : BLACK;
    BORDER-BOTTOM : 1px solid #999999;
    BACKGROUND-COLOR : white
}
09 A:link { text-decoration:nonecolor:#696969 }
10 A:hover { text-decoration:yescolor:#66CCFF }
11 A:visited { text \decoration:nonecolor:#330066 }

```

- 예제 JSP

WebContent/member/useJDBC.jsp

```

01 <%@ page contentType="text/html;charset=EUC-KR" import="java.sql.*" %>
02 <%
03     Class.forName ("oracle.jdbc.driver.OracleDriver");
04     Connection conn = null;
05     Statement stmt = null;
06     ResultSet rs = null ;
07     String id = "",
08     passwd = "",
09     name = ""
10    mem_num1 = "",
11    mem_num2 = "",
12    e_mail = "",
13    phone = "",
14    zipcode = "",
15    address = "",
16    job = "";
17    int counter = 0;
18    try {
19        conn = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "scott", "tiger");
20        stmt = conn.createStatement( );
21        rs = stmt.executeQuery("SELECT * FROM MEMBER");
22    %>
23    <html>
24    <head>
25    <title>JSP에서 데이터베이스 연동</title>
26
27    <link href="style.css" rel="stylesheet" type = "text/css">
28    </head>
29    <body bgcolor="#FFFFCC">
30        <h2>JSP 스크립틀릿에서 데이터베이스 연동 예제</h2><br>
31        <h3>회원정보</h3>
32        <table bordercolor="#0000ff" border="1">
33            <tr>
34                <td> <strong>ID</strong> </td>

```

```

35   <td> < strong> PASSWD </strong></td>
36   <td> <strong>NAME</strong> </td>
37   <td> <strong>MEM_NUM1</strong></td>
38   <td> <strong>MEM_NUM2</strong></td>
39   <td><strong>E_MAIL</strong></td>
40   <td><strong>PHONE</strong></td>
41   <td><strong>ZIPCODE/ADDRESS</strong> </td>
42   <td><strong>JOB</strong></td>
43   </tr>
44 <%
45   if( rs!=null ) {
46       while( rs.next( ) ) {
47           id = rs.getString("id");
48           passwd = rs.getString("passwd");
49           name = rs.getString("name");
50           mem_num1 = rs.getString( "mem_num1");
51           mem_num2 = rs.getString( "mem_num2 ");
52           e_mail = rs.getString("e_mail");
53           phone = rs.getString( "phone");
54           zipcode = rs.getString("zipcode");
55           address = rs.getString( "address");
56           job = rs.getString("job");
57       }%>
58   <tr>
59       <td><%= id %></td>
60       <td><%= passwd %></td>
61       <td><%= name %></td>
62       <td><%= mem_num1 %></td>
63       <td><%= mem_num2 %></td>
64       <td><%= e_mail %></td>
65       <td><%= phone %></td>
66       <td><%= zipcode %><%= address %></td>
67       <td><%= job %></td>
68   <%

```

```
69         counter++;
70     } //end while
71 } //end if
72 %>
73     </tr>
74 </table><br>
75 total records : <% = counter %>
76 <%
77 } catch( SQLException sqlException ) {
78     System.out.println("sql exception");
79 } catch( Exception exception ) {
80     System.out.println( "exception");
81 } finally {
82     if(rs != null )
83         try { rs.close( ); }
84         catch( SQLException ex ) { }
85     if( stmt != null )
86         try { stmt.close( ); }
87         catch( SQLException ex ) { }
88     if( conn != null )
89         try{ conn.close( ); }
90         catch( Exception ex ){ }
91     }
92 %>
93 </body>
94 </html>
```

Chapter9. 파일 업로드

1 COS 라이브러리

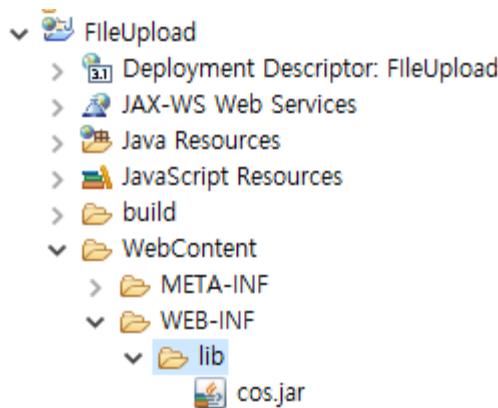
JSP에서 자주 사용되는 기능인 파일 업로드는 Java에서 기본으로 지원하지 않지만, cos.jar 오픈 라이브러리를 이용하면 쉽게 구현 가능할 수 있다.

- COS 라이브러리를 얻을 수 있는 주소

```
http://servlets.com/cos/
```

웹 페이지 하단에서 볼 수 있는 Download 항목에서 cos파일을 다운 받는다.

압축 파일 내 lib폴더의 cos.jar 파일을 복사하여 자신의 프로젝트의 WebContent/WEB-INF/lib 폴더에 붙여 넣는다.



- 파일 업로드를 위한 form 태그

```
<form name = "사용자정의" method = "post" enctype = "multipart/formdata">  
    <input type = "file" name = "사용자정의">  
</form>
```

method를 post로 지정

enctype 속성을 multipart/formdata로 추가

input 태그의 type 을 file로 지정

enctype 속성을 multipart/formdata는 서버로 데이터가 전송될 때 일반 텍스트뿐만 아니라 파일 형식의 데이터도 전송됨을 알리기 위해 지정하는 것이다. 이를 추가하지 않으면 file 선택 박스에서 선택된 파일의 이름만 텍스트로 전송되고 파일 객체가 서버로 전송되지 않는다.

또한 파일을 전송할 경우 form으로 전송하는 데이터의 크기가 커지므로 get 메소드 방식으로

전송하는 것이 불가능해진다.

2 파일 업로드 하기

- 파일 전송 폼 화면

WebContent/upload.jsp

```
01 <%@ page language="java" contentType="text/html; charset=UTF-8"
02     pageEncoding="UTF-8"%>
03 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
04 "http://www.w3.org/TR/html4/loose.dtd">
05 <html>
06 <head>
07 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
08 <title>Insert title here</title>
09 </head>
10 <body>
11
12 <form action="upload.do" method="post" enctype="multipart/form-data">
13
14 글쓴이 : <input type="text" name="name"/><br/>
15 제 목 : <input type="text" name="title"/><br/>
16 첨부파일 : <input type="file" name="uploadFile"/><br/>
17 <input type="submit" value="전송"/>
18 </form>
19
20 </body>
21 </html>
```

- 파일 업로드를 위한 서블릿 클래스

src/controller/UploadServlet.java

```
01 package controller;
02
03 import java.io.IOException;
04 import java.io.PrintWriter;
```

```
05  
06 import javax.servlet.ServletContext;  
07 import javax.servlet.ServletException;  
08 import javax.servlet.annotation.WebServlet;  
09 import javax.servlet.http.HttpServlet;  
10 import javax.servlet.http.HttpServletRequest;  
11 import javax.servlet.http.HttpServletResponse;  
12  
13 import com.oreilly.servlet.MultipartRequest;  
14 import com.oreilly.servlet.multipart.DefaultFileRenamePolicy;  
15  
16 @WebServlet("/upload.do")  
17 public class UploadServlet extends HttpServlet {  
18     private static final long serialVersionUID = 1L;  
19  
20     protected void doPost(  
21         HttpServletRequest request,  
22         HttpServletResponse response) throws ServletException, IOException {  
23  
24         request.setCharacterEncoding("UTF-8");  
25         response.setContentType("text/html; charset=UTF-8");  
26  
27         PrintWriter out = response.getWriter();  
28  
29         // 파일 저장 경로  
30         String fileSavePath = "upload";  
31  
32         // 파일 크기 10M 제한  
33         int uploadSizeLimit = 10 * 1024 * 1024;  
34         String encType = "UTF-8";  
35  
36         ServletContext context = getServletContext();  
37  
38         String uploadPath = context.getRealPath(fileSavePath);
```

```

39     System.out.println(uploadPath);
40
41     MultipartRequest multi = new MultipartRequest(
42         request, // request 객체
43         uploadPath, // 서버 상 업로드 될 디렉토리
44         uploadSizeLimit, // 업로드 파일 크기 제한
45         encType, // 인코딩 방법
46         new DefaultFileRenamePolicy() // 동일 이름 존재 시 새로운 이름 부여 방식
47     );
48
49     // 업로드 된 파일 이름 얻어오기
50     String file = multi.getFilesystemName("uploadFile");
51
52     if( file == null ) {
53         // 업로드 실패 시
54         System.out.println("업로드 실패");
55     } else {
56         // 업로드 성공 시
57         out.println("<br> 글쓴이 :" + multi.getParameter("name"));
58         out.println("<br> 제목 :" + multi.getParameter("title"));
59         out.println("<br> 첨부파일명 :" + file);
60     }
61 }
62 }
```

업로드된 파일은 서버의 지정 폴더에 저장된다.

eclipse에서 서버를 구동하여 구현한 경우

{workspace}\metadata\plugins\org.eclipse.wst.server.core\tmp0\wtpwebapps\fileUpload\upload

에서 확인할 수 있다.

MultipartRequest 클래스는 cos.jar 라이브러리에서 제공되는 파일 업로드를 담당하는 클래스이다.
파일 업로드를 담당하는 생성자와 메소드를 포함한다.

- 파일 업로드를 하기 위한 import

```

import com.oreilly.servlet.MultipartRequest;
import com.oreilly.servlet.multipart.DefaultFileRenamePolicy;

```

- MultipartRequest 생성자

매개 변수	설명
request	MultipartRequest 객체와 연결될 request 객체
saveDirectory	서버 측에 저장될 경로
maxPostSize	업로드 최대 크기
encoding	파일 인코딩 방식 (파일 이름이 한글일 경우 UTF-8 지정)
policy	파일 중복 처리를 어떻게 할지 결정 (new DefaultFileRenamePolicy() 를 이용해 간단히 지정할 수 있다. 파일 명 뒤에 숫자를 붙인다.)

- MultipartRequest의 메소드

메소드 명	설명
getParameterNames()	폼에서 전송된 파라미터 열거체로 리턴
getParameterValues()	폼에서 전송된 파라미터를 배열로 리턴
getParameter()	객체의 해당 파라미터 값을 가져옴
getFileName()	파일을 여러 개 업로드 할 경우 값들을 열거체로 리턴
getFilenames()	서버에 실제로 업로드된 파일 이름
getOriginalFileName()	클라이언트가 업로드한 원본 파일 이름
getContentType()	업로드 파일의 컨텐트 타입 확인
getFile()	서버에 업로드 된 파일 정보를 객체로 얻어옴

3 한꺼번에 여러 파일 업로드하기

- 파일 전송 폼 화면

WebContent/multiUpload.jsp	
01	<%@ page language="java" contentType="text/html; charset=UTF-8"
02	pageEncoding="UTF-8"%>
03	<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
04	"http://www.w3.org/TR/html4/loose.dtd">
05	<html>

```
06 <head>
07 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
08 <title>Insert title here</title>
09 </head>
10 <body>
11
12 <form action="multiUpload.do" method="post" enctype="multipart/form-data">
13 첨부파일1 : <input type="file" name="uploadFile1"/><br/>
14 첨부파일2 : <input type="file" name="uploadFile2"/><br/>
15 첨부파일3 : <input type="file" name="uploadFile3"/><br/>
16 <input type="submit" value="전송"/>
17 </form>
18
19 </body>
20 </html>
```

- 파일 업로드를 위한 서블릿 클래스

src/controller/MultiUploadServlet.java

```
01 package controller;
02
03 import java.io.IOException;
04 import java.io.PrintWriter;
05 import java.util.Enumeration;
06
07 import javax.servlet.ServletContext;
08 import javax.servlet.ServletException;
09 import javax.servlet.annotation.WebServlet;
10 import javax.servlet.http.HttpServlet;
11 import javax.servlet.http.HttpServletRequest;
12 import javax.servlet.http.HttpServletResponse;
13
14 import com.oreilly.servlet.MultipartRequest;
15 import com.oreilly.servlet.multipart.DefaultFileRenamePolicy;
```

```
17 @WebServlet("/multiUpload.do")
18 public class MultiUploadServlet extends HttpServlet {
19     private static final long serialVersionUID = 1L;
20
21     protected void doPost(
22         HttpServletRequest request,
23         HttpServletResponse response) throws ServletException, IOException {
24
25         request.setCharacterEncoding("UTF-8");
26         response.setContentType("text/html"); charset=UTF-8");
27
28         PrintWriter out = response.getWriter();
29
30         // 파일 저장 경로
31         String fileSavePath = "upload";
32
33         // 파일 크기 10M 제한
34         int uploadSizeLimit = 10 * 1024 * 1024;
35         String encType = "UTF-8";
36
37         ServletContext context = getServletContext();
38         String uploadPath = context.getRealPath(fileSavePath);
39         // System.out.println(uploadPath);
40
41         MultipartRequest multi = new MultipartRequest(
42             request, // request 객체
43             uploadPath, // 서버 상 업로드 될 디렉토리
44             uploadSizeLimit, // 업로드 파일 크기 제한
45             encType, // 인코딩 방법
46             new DefaultFileRenamePolicy() // 동일 이름 존재 시 새로운 이름 부여 방식
47         );
48
49         Enumeration files = multi.getFileNames();
50
```

```
51 while( files.hasMoreElements() ) {  
52     // 업로드 된 파일 이름 얻어오기  
53     String file = (String) files.nextElement();  
54     String fileName = multi.getfilesystemName(file);  
55  
56     out.println("<br> 글쓴이 :" + multi.getParameter("name"));  
57     out.println("<br> 제목 :" + multi.getParameter("title"));  
58     out.println("<br> 첨부파일명 :" + file);  
59 }  
60  
61 }  
62 }
```

Chapter10. Ajax

1 Ajax, Asynchronous JavaScript + XML

XML기반의 기술로 비동기적 자바스크립트 XML의 준말이다. 자바 스크립트로 HTTP 요청을 보내 XML로 응답을 받는 기술이다.

표준HTML과 CSS로 작성된 화면과 DOM(Document Object Model)을 이용한 상호작용, 서버와의 데이터 교환은 XML형식을 이용하며 비동기 통신에 웹브라우저에 내장된 XMLHttpRequest 객체를 이용하는 자바스크립트 기술이다.

기존의 방식의 웹 사이트는 웹 브라우저가 웹 서버에 요청을 전송하면 웹 서버는 JSP/PHP/ASP 등의 언어로 이루어진 서버 측 어플리케이션을 사용해 요청을 처리한 뒤, 처리 결과를 HTML로 생성해 웹 브라우저에 응답한다. 웹 브라우저는 응답받은 HTML을 분석한 뒤 그 내용을 화면에 그려 주게 된다. 즉, 웹 브라우저가 웹 서버와 통신을 하고 요청 결과는 HTML로 생성되며 사용자 입장에서 페이지 이동이 발생하는 것을 확인할 수 있게 된다.

Ajax 방식의 웹 사이트는 별도의 웹 페이지 호출 없이 웹 어플리케이션과 비동기적으로 데이터를 주고 받은 후 클라이언트에서 응답 받은 데이터를 처리할 수 있게 된다. 웹 페이지의 이동없이 화면을 유지한 채 웹 서버와 데이터 교환을 할 수 있게 된다.

Ajax 방식의 웹 사이트는 기존의 방식과는 다르게 XMLHttpRequest 객체가 웹 서버와 통신을 하며, 응답 결과가 HTML이 아닌 XML 또는 단순 텍스트로 처리된다.

1.1 Ajax 방식 웹 사이트의 장점

- 비동기 통신 기반으로 데이터 전송과는 별개로 사용자는 다른 작업을 수행할 수 있다
- 데이터만을 응답으로 받으므로서 처리 속도가 빠르며 전송 데이터의 크기도 적어진다
- 불필요한 데이터 요청을 최소화할 수 있다
- 많은 부분을 웹 브라우저에서 처리되므로서 서버에 부하를 줄일 수 있다

2 Ajax의 주요 구성요소

2.1 XMLHttpRequest

웹 서버와 통신을 담당한다

사용자의 요청을 웹 서버에 전송하고, 웹 서버로부터 받은 결과 데이터를 웹 브라우저에 전달

2.2 DOM, Document Object Model

문서의 구조를 나타낸다

폼 등의 정보나 화면 구성을 조작할 때 사용할 수 있다

2.3 CSS

화면 구성요소의 스타일과 관련된 부분을 담당한다

2.4 JavaScript

HTML 문서의 상호작용 처리를 담당한다

XMLHttpRequest 객체를 사용해서 웹 서버에 요청을 전송하며, 응답이 오면 DOM, CSS 등을 사용해서 화면을 조작할 수 있다

3 XMLHttpRequest 객체(XHR 객체)

Ajax통신을 하기 위한 객체

3.1 XMLHttpRequest 객체의 메소드

3.1.1 void open(String method, String url, boolean asynch)

요청을 초기화 한다

method : HTTP 메소드 지정

url : 접속할 URL 지정, 웹 브라우저 보안 상 접속할 URL은 현재페이지와 같은 도메인이어야 한다

asynch : 동기(false), 비동기(true) 지정

method 와 url 파라미터는 필수이며 asynch는 필수가 아니다. 동기화 여부는 Ajax의 이점을 활용하기 위해 기본적으로 비동기(true)로 설정하는 것이 좋다. 만약 동기(false)로 설정한다면 요청에 대해 동기처리를 하므로 서버의 응답을 받을 때까지 대기상태가 된다.

3.1.2 void send()

서버에 요청을 보낸다

3.2 XMLHttpRequest 객체 속성

3.2.1 onreadystatechange

자바스크립트 콜백 함수를 저장

해당 콜백 함수는 readyState 값이 변경될 때마다 호출된다

3.2.2 readyState

요청의 상태를 의미하는 값

- 0 : UNINITIALIZED, 객체만 생성되고 초기화되지 않은 상태(open() 메소드가 호출 전)
- 1 : LOADING, open메서드가 호출되고 아직 send 메소드가 호출 되지 않은 상태
- 2 : LOADED, send()는 호출했지만 states와 header는 도착하지 않은 상태
- 3 : INTERACTIVE, 데이터의 일부만 받은 상태
- 4 : COMPLETED, 데이터를 전부 받은 상태

3.2.3 responseText

서버의 응답을 String 형식으로 나타냄

3.2.4 responseXML

서버의 응답을 XML 형식으로 나타냄

3.2.5 status

서버로부터 받은 HTTP 응답 상태 코드

3.2.6 statusText

HTTP 상태코드에 대한 텍스트

4 XHR 객체 생성하기

Ajax 통신을 하기 위해 XMLHttpRequest 객체를 생성해야 한다. IE 7.0 이전에는 모질라 웹 브라우저와 IE 브라우저가 각각 XHR 객체를 생성하는 방식이 달랐지만 IE 7.0 부터는 동일하게 new XMLHttpRequest()를 통해 객체를 생성할 수 있다.

```
function getXMLHttpRequest() {
```

```
// IE에서 XMLHttpRequest 객체 구하기
if( window.ActiveXObject ) {
    try {
        // IE 최신 버전
        return new ActiveXObject("Msxml2.XMLHTTP");
    } catch(e) {
        try {
            // IE 이전 버전
            return new ActiveXObject("Microsoft.XMLHTTP");
        } catch(e1) {
            return null;
        }
    }
} else if(window.XMLHttpRequest) {
    //IE이외의 파이어폭스, 오페라와 같은 브라우저에서 XMLHttpRequest 객체구하기
    return new XMLHttpRequest();
} else {
    return null;
}
```

5 웹 서버에 요청

5.1 GET 방식

```
httpRequest = getXMLHttpRequest( );
httpRequest.open("GET", "/test.jsp?id=admin&pw=1234", true);
httpRequest.send(null);
```

5.2 POST 방식

```
httpRequest = getXMLHttpRequest( );
httpRequest.open("POST", url, true);
httpRequest.send("id=admin&pw=1234");
```

6 웹 서버 응답 반영하기

XMLHttpRequest 객체의 속성인 onreadystatechange를 사용하여 콜백 함수를 지정하고 웹 서버로부터 응답이 도착하면 해당 콜백 함수가 불러와지도록 설정할 수 있다.

- GET 방식의 요청에 응답이 왔을 때 콜백 함수 호출되도록 하기

```
httpRequest = getXMLHttpRequest();
httpRequest.onreadystatechange = callbackFunc;
httpRequest.open("GET", "/test.jsp?id=admin&pw=1234", true);
httpRequest.send(null);

function callbackFunc() {
    // 서버의 응답에 알맞은 작업 수행
}
```

콜백함수의 작성은 XHR 객체의 status 속성을 확인해 정상적으로 수행하거나 에러메시지를 보여주도록 작성할 수 있다.

7 XMLHttpRequest 모듈 만들기

XMLHttpRequest 객체를 이용하여 Ajax 방식으로 코드를 구현하려면 매 페이지마다 반복되는 코드가 존재하게 된다. 이를 보다 편하게 사용하기 위해 js 파일을 만들어 모듈화 해보도록 한다.

- XMLHttpRequest 모듈 : httpRequest.js

```
httpRequest.js

01 //XMLHttpRequest 객체를 생성
02 function getXMLHttpRequest() {
03     if (window.ActiveXObject) {
04         try {
05             return new ActiveXObject("Msxml2.XMLHTTP");
06         } catch(e) {
07             try {
08                 return new ActiveXObject("Microsoft.XMLHTTP");
09             } catch(e1) { return null; }
10         }
11     } else if (window.XMLHttpRequest) {
```

```
12     return new XMLHttpRequest();
13 } else {
14     return null;
15 }
16 }
17
18 //생성된 XMLHttpRequest 객체 전역변수
19 var httpRequest = null;
20
21 //지정 메소드(GET/POST), 지정 URL, 파라미터 값을 사용하여 웹 서버에 요청을 전송
22 function sendRequest(method, url, params, callback) {
23     httpRequest = getXMLHttpRequest();
24
25     var httpMethod = method ? method : 'GET';
26     if (httpMethod != 'GET' && httpMethod != 'POST') {
27         httpMethod = 'GET';
28     }
29
30     var httpParams = (params == null || params == "") ? null : params;
31     var httpUrl = url;
32
33     // GET 메소드면 URL 뒤에 파라미터를 붙임
34     if (httpMethod == 'GET' && httpParams != null) {
35         httpUrl = httpUrl + "?" + httpParams;
36     }
37
38     // 비동기식으로 XMLHttpRequest 객체 사용
39     httpRequest.open(httpMethod, httpUrl, true);
40
41     // 웹 서버에 전송할 컨텐트 타입 지정
42     httpRequest.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
43
44     // 콜백 함수 지정
45     httpRequest.onreadystatechange = callback;
```

```

46
47 // HTTP 요청 방식이 'POST'면 send() 함수를 통해 파라미터 전송
48 httpRequest.send(httpMethod == 'POST' ? httpParams : null);
49 }

```

- 화면담당 웹 페이지 : ajaxTest.jsp

ajaxTest.jsp

```

01 <%@ page language="java" contentType="text/html; charset=UTF-8"
02             pageEncoding="UTF-8"%>
03 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
04 "http://www.w3.org/TR/html4/loose.dtd">
05 <html>
06 <head>
07 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
08 <title>안녕하세요</title>
09
10 <!-- XMLHttpRequest 모듈 로딩 -->
11 <script type="text/javascript" src="httpRequest.js"></script>
12 <script type="text/javascript">
13
14 // [입력]버튼 클릭시 호출.
15 // sendRequest() 함수를 사용하여 폼에 입력한 값을 hello.jsp에 전송
16 function helloToServer() {
17     var params = "name="+encodeURIComponent(document.f.name.value);
18     sendRequest("POST", "hello.jsp", params, helloFromServer);
19 }
20
21 //웹 서버로부터 올바른 응답 결과가 도착하면 hello.jsp가 생성한 텍스트 출력
22 function helloFromServer() {
23     if (httpRequest.readyState == 4) {
24         if (httpRequest.status == 200) {
25             alert("서버응답:"+httpRequest.responseText);
26         }
27     }

```

```

28 }
29
30 </script>
31
32 </head>
33 <body>
34
35 <form name="f">
36   <input type="text" name="name" />
37   <input type="button" value="입력" onclick="helloToServer()" />
38 </form>
39
40 </body>
41 </html>

```

- 요청/응답 페이지 : hello.jsp

hello.jsp	
01	<%@ page contentType="text/html; charset=UTF-8"%>
02	<%
03	// UTF-8 인코딩으로 파라미터 읽음
04	request.setCharacterEncoding("UTF-8");
05	String name = request.getParameter("name");
06	%>
07	안녕하세요, <%=name %> 님!

8 jQuery Ajax

Ajax는 브라우저마다 다른 코드를 가지게 되는 단점이 있다. 하지만 jQuery가 제공해주는 Ajax 기능을 사용하면 크로스브라우징을 제공하기 때문에 한가지 코드로 다양한 브라우저에 대응할 수 있게 된다.

8.1 jQuery의 Ajax API

Ajax 통신에 필요한 여러가지 설정에 관련된 기능을 제공하는 Global Ajax Event Headers, 통신에 는 직접적으로 관여하지 않지만 도움이 되는 Helper Functions, POST와 GET 메소드 통신을 할 수

있도록 하는 Shorthand Methods, 가장 기본적인 통신을 제공하는 Low-Level Interface 등을 제공한다. 이처럼 jQuery는 Ajax 관련 다양한 API를 제공하지만 그 중 Low-Level Interface API를 많이 사용한다.

- jQuery Ajax API

```
http://api.jquery.com/category/ajax/
```

8.2 jQuery Ajax Low-Level Interface API

- jQuery.ajax API 의 기본 형태

```
jQuery.ajax( url [, settings ] )
```

url : Ajax 통신 요청을 보낼 url

settings : 통신을 위한 여러 설정

- jQuery.ajax API 기본 예제

```
$ajax({  
    url : "ajaxData.jsp?type=1",  
    dataType : "text",  
    type : "GET",  
    success : function(data){  
        alert("통신데이터 값 :" + data);  
        $("#resultArea").html(data);  
    }  
    error : function( ){  
        alert('실패!!');  
    },  
});
```

url : ajax 요청 주소

data : 서버로 데이터를 전송할 옵션

dataType : 서버측에서 데이터를 해석할 형식 (xml, json, script, html 등)

success : 요청에 대한 처리가 성공했을 때 호출할 콜백

type : HTTP 전송 메소드 지정 (get, post)

8.3 ajax 메소들 이용한 간단한 계산기 구현

- 계산기 화면

calcForm.jsp

```
01 <%@ page contentType="text/html; charset=UTF-8"%>
02 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
03 "http://www.w3.org/TR/html4/loose.dtd">
04 <html>
05 <head>
06 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
07 <title>Insert title here</title>
08
09 <script type="text/javascript" src="http://code.jquery.com/jquery-1.12.4.min.js"></script>
10
11 <script type="text/javascript">
12 $(function(){
13     $("#btnSend").click(function(){
14         var num1=$("#num1").val();
15         var num2=$("#num2").val();
16         var oper=$("#oper").val();
17
18         var url="calc.jsp";
19
20         var query="n1="+num1+"&n2="+num2;
21
22         query += "&oper=" + oper;
23
24         $.ajax({
25             type:"POST"
26             , url:url
27             , data:query
28             , success:function(data) {
29                 $("#result").html(data);
30                 $("#num1").val("");
31                 $("#num2").val("");
```

```

32     $("#oper").val("add");
33 }
34 , error:function(e) {
35     alert(e.responseText);
36 }
37 });
38 });
39 );
40
41 </script>
42 </head>
43 <body>
44
45 <input type="text" id="num1">
46 <select id="oper">
47     <option value="add">더하기</option>
48     <option value="sub">빼기</option>
49     <option value="mul">곱하기</option>
50     <option value="div">나누기</option>
51 </select>
52 <input type="text" id="num2">
53 <input type="button" value=" = " id="btnSend"><br>
54
55 <hr>
56 <div id="result"></div>
57
58 </body>
59 </html>

```

- 계산 페이지

calc.jsp

01	<%@ page contentType="text/html; charset=UTF-8"%>
02	
03	<%

```
04 request.setCharacterEncoding("utf-8");
05
06 int num1 = Integer.parseInt(request.getParameter("n1"));
07 int num2 = Integer.parseInt(request.getParameter("n2"));
08 String oper=request.getParameter("oper");
09
10 String s="";
11 if(oper.equals("add")) {
12     s = String.format("%d+%d=%d", num1, num2, num1+num2);
13 } else if(oper.equals("sub")) {
14     s = String.format("%d-%d=%d", num1, num2, num1-num2);
15 } else if(oper.equals("mul")) {
16     s = String.format("%d*%d=%d", num1, num2, num1*num2);
17 } else if(oper.equals("div")) {
18     s = String.format("%d/%d=%d", num1, num2, num1/num2);
19 }
20 %>
21
22 <b><%=s%></b>
```

Chapter11. 모델2 기반의 MVC 패턴

1 MODEL 1 방식의 웹 개발

디자인코드인 HTML과 비즈니스 로직인 자바코드를 따로 구분하지 않고 JSP파일 내에 함께 기술하는 방식을 말한다. 하나의 JSP 파일 내에 로직 처리를 위한 자바코드와 출력을 위한 코드를 함께 섞어서 작성한다. 모델 1 방식의 개발 형태를 보면 JSP/Servlet만으로 개발 된 경우, JSP + Java Bean을 이용한 경우, JSP + Custom Tag를 이용한 경우, 이 모든 것을 섞어 개발한 경우가 있다. 과거에 많이 사용되었던 웹 개발 아키텍처이고 또한 현재에도 많은 개발자들이 사용하고 있으며 간단한 페이지를 구성할 때 어울리는 방식이다.

하지만 JSP페이지가 너무 복잡해지는 경우고 있고 개발자와 디자이너 간의 의사소통이 어려운 점 등 때문에 모델 2 개발 방식을 많이 취하게 된다.

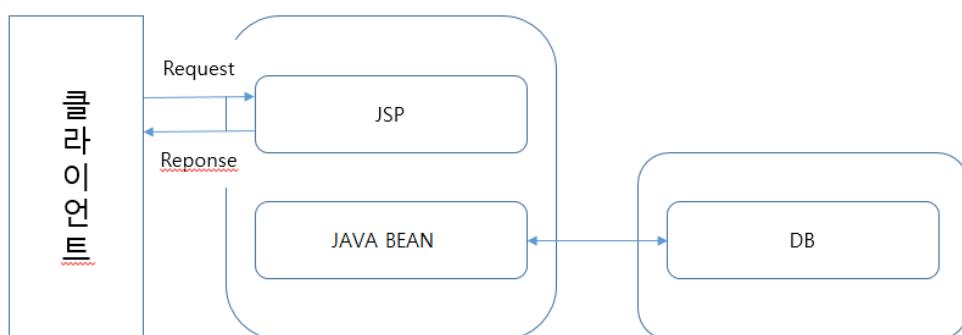
1.1 모델 1 방식의 장점

- 개발 속도가 빠르다
- 구조가 단순하여 익히기 쉽다
- 개발자의 스킬이 낮아도 배우기 쉽고 빠르게 적용 가능하다
- 중소형 프로젝트에 적합하다

1.2 모델 1 방식의 단점

- 표현 코드(디자인)과 비즈니스 로직이 하나의 JSP페이지에 표현되므로 코드가 복잡해진다
- 하나의 코드에 디자인코드와 로직코드가 혼재하게 되어 개발자와 디자이너의 분리 작업이 힘들어 진다
- JSP파일이 복잡해짐으로 유지보수가 어려워진다
- 비즈니스 로직의 재사용성이 떨어진다

1.3 모델 1 방식의 구조



2 MODEL 2 방식의 웹

웹 어플리케이션을 개발할 때 MVC패턴을 적용하여 웹 어플리케이션의 개발이 가능하도록 구현한 것이다. MVC는 Model, View, Controller로 각각의 역할을 나누어 작업하고자 하는 일을 분담시키는 것을 말한다.

Model은 어플리케이션 로직을 담당하는 부분으로 데이터베이스와의 로직을 담당하는 부분을 말한다.

View는 client에게 보여지는 부분이며 Model에서 생성된 Data를 client에게 제공하는 역할을 담당한다.

Controller는 사용자의 요청을 받아서 요청에 해당하는 비즈니스 로직을 수행하도록 하고, 작업 결과에 따라 응답을 결정하는 역할을 한다. Model과 View사이의 데이터 전달 역할을 한다.

모델 2 웹 개발 방식은 MVC 구조를 웹에 적용하여 개발하는 방식을 뜻한다.

View는 JSP가 담당하고, Controller는 Servlet이 담당하며, Model은 JavaBean이나 서비스 클래스를 이용하여 담당하게 한다.

2.1 모델 2 방식의 장점

- 출력을 위한 뷰 코드와 로직 처리를 위한 자바 코드를 분리하기 때문에 코드가 덜 복잡해진다
- 뷰, 로직에 대한 명확한 역할 분담이 가능하다
- UI 레이어(화면 구현)을 단순화 시킴으로서 디자이너의 작업을 수월하게 한다
- 기능에 따라 분리되어 있어 유지보수가 쉬워진다
- 비즈니스 로직의 재사용성이 높아지고 확장성이 용이하다

2.2 모델 2 방식의 단점

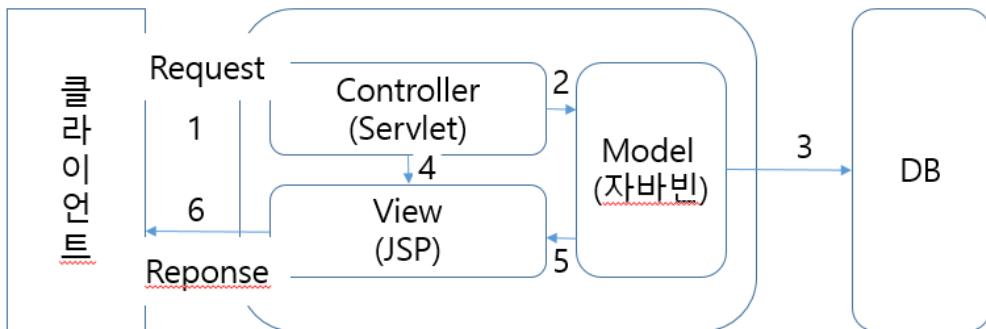
- 구조가 복잡해져 기술 습득이 어렵고 작업량이 많아진다
- 자바 코드에 대한 비교적 높은 이해도가 요구된다
- 구조 설계를 위한 시간이 많이 소요된다 (프로젝트 초반 개발속도의 저하 초래)

2.3 모델 1과 모델2

모델2의 웹 개발 방식은 큰 규모의 프로젝트나 업데이트가 빈번한 프로젝트에는 장점이 크게 적용되지만 소규모 프로젝트나 규모가 작은 프로젝트일 경우 복잡한 형태 때문에 작업량만 늘어

날 수 있다. 이럴 경우 모델1 방식으로 개발하는 것이 더 나을 수도 있다.

2.4 모델 2방식의 구조



3 모델2 방식의 구성 및 흐름

Controller는 중앙에 위치하여 View와 Model사이의 연동을 담당한다. 사용자의 요청은 Controller에서 받으며 Model을 통한 비즈니스 로직을 처리하고 수행 결과를 View를 통해 사용자에게 보여지게 한다. Controller 역할을 하는 서블릿이 매우 중요하며, 서블릿을 잘못 개발할 경우 웹 어플리케이션에 미치는 영향이 매우 크다.

3.1 Controller 역할

- 사용자의 요청을 받는다
- 사용자의 요청을 분석한다
- 사용자의 요청을 처리할 자바 빈을 생성하고, 비즈니스 로직이 구현된 메소드를 실행한다
- 비즈니스 로직 수행 후 사용자의 요청을 JSP페이지나 혹은 특정 URL로 이동시킨다

3.2 Model 역할

- 비즈니스 로직이 구현되어 있다
- Database에 데이터를 추가, 조회, 변경, 조회하는 작업을 수행한다
- View에 제공할 Data를 가공한다

3.3 View 역할

- 클라이언트에게 최종적으로 보여지는 영역이다

- JSP와 JSTL을 이용하여 구현한다

4 요청 파라미터를 명령어로 전달하는 방법에 의한 Model2 구현

4.1 필요 클래스

- ControlServlet : controller의 역할을 수행하는 서블릿
- ActionFactory : 사용자의 요청을 처리를 담당하는 비즈니스 로직이 구현된 XXXAction 객체의 생성하는 역할을 수행한다
- Action : 모든 XXXAction 클래스가 implements할 인터페이스. 비즈니스 로직을 재정의할 메소드를 선언한다
- XXXAction : Action 인터페이스의 메소드를 재정의하고 있는 클래스. 실질적인 비즈니스 로직의 구현체이다
- ActionForward : XXXAction의 비즈니스 로직 수행 후 ControlServlet에게 반환하는 객체 (이동할 url과 이동방법을 저장하고 있음)

비즈니스 로직의 수행이 필요한 경우 반드시 ControlServlet이 해당 요청을 처리할 수 있도록 클라이언트의 요청 url을 test.do의 형태로 하고, web.xml에 매팅 정보를 추가한다.

sdfsdf

http://localhost/model2/test.do?cmd=list

http://localhost/model2/test.do?cmd=insert

위와 같은 형태로 호출될 수 있도록 하여야 한다.

4.2 ActionForward.java

src/controller/ActionForward.java

```
01 package controller;  
02  
03 public class ActionForward {  
04     private String url;  
05     private boolean redirect;  
06 }
```

```
07 public ActionForward() {
08 }
09 public ActionForward(String url) {
10     this.url = url;
11 }
12 public ActionForward(String url, boolean redirect) {
13     this.url = url;
14     this.redirect = redirect;
15 }
16
17 public boolean isRedirect() {
18     return redirect;
19 }
20
21 public void setRedirect(boolean redirect) {
22     this.redirect = redirect;
23 }
24
25 public String getUrl() {
26     return url;
27 }
28
29 public void setUrl(String url) {
30     this.url = url;
31 }
32 }
```

4.3 Action.java

src/action/Action.java

```
01 package action;
02
03 import controller.ActionForward;
04
05 import java.io.IOException;
```

```
06 import javax.servlet.http.HttpServletRequest;
07 import javax.servlet.http.HttpServletResponse;
08
09 public interface Action {
10     public ActionForward execute(HttpServletRequest request,
11                                 HttpServletResponse response) throws IOException;
12 }
```

4.4 IndexAction.java

src/action/IndexAction.java

```
01 package action;
02
03 import controller.ActionForward;
04
05 import java.io.IOException;
06 import javax.servlet.http.HttpServletRequest;
07 import javax.servlet.http.HttpServletResponse;
08
09 public class IndexAction implements Action {
10     public ActionForward execute(HttpServletRequest request,
11                                 HttpServletResponse response) throws IOException {
12         System.out.println("IndexAction의 execute() 수행됨!");
13         return new ActionForward("index.jsp", false);
14     }
15 }
```

4.5 ActionFactory.java

src/controller/ActionFactory.java

```
01 package controller;
02
03 import action.Action;
04 import action.IndexAction;
05
```

```
06 public class ActionFactory {  
07     private static ActionFactory factory;  
08     private ActionFactory() {}  
09  
10    public static synchronized ActionFactory getInstance() {  
11        if(factory == null) {  
12            factory = new ActionFactory();  
13        }  
14  
15        return factory;  
16    }  
17    public Action getAction(String cmd) {  
18        Action action = null;  
19  
20        if(cmd.equals("index")) {  
21            action = new IndexAction();  
22        }  
23  
24        return action;  
25    }  
26 }
```

4.6 ControllerServlet.java

src/controller/ControllerServlet.java

```
01 package controller;  
02  
03 import action.Action;  
04  
05 import java.io.IOException;  
06 import java.io.PrintWriter;  
07  
08 import javax.servlet.RequestDispatcher;  
09 import javax.servlet.ServletException;  
10 import javax.servlet.http.HttpServlet;
```

```
11 import javax.servlet.http.HttpServletRequest;
12 import javax.servlet.http.HttpServletResponse;
13
14 public class ControllerServlet extends HttpServlet {
15     private static final long serialVersionUID = 1L;
16
17     @Override
18     public void service(HttpServletRequest request,
19             HttpServletResponse response) throws ServletException, IOException {
20         request.setCharacterEncoding("utf-8");
21
22         String cmd = request.getParameter("cmd");
23
24         if(cmd != null) {
25             ActionFactory factory = ActionFactory.getInstance( );
26             Action action = factory.getAction(cmd);
27             ActionForward af = action.execute (request, response);
28
29             if(af.isRedirect()) {
30                 response.sendRedirect(af.getUrl());
31             } else {
32                 RequestDispatcher rd = request.getRequestDispatcher(af.getUrl());
33                 rd.forward(request, response);
34             }
35         } else {
36             response.setContentType("text/html;charset=euc—kr");
37             PrintWriter out = response.getWriter();
38             out .println ("<html>");
39             out.println("<head> <title>Error</title> </head>");
40             out.println("<body>");
41             out.println("<h4>올바른 요청이 아닙니다!</h4>");
42             out .println("<h4>http://localhost:8080/mvc/test.do?cmd=요청키워드 </h4>");
43             out .println("</body>");
44             out .println("</html>");
```

```
45    }
46    }
47 }
```

4.7 index.jsp

WebContent/index.jsp

```
01 <%@ page contentType="text/html;charset=utf-8"%>
02 <html>
03 <head>
04 <title>MVC 패턴</title>
05 </head>
06 <body>
07 <h3 align="center">index page 입니다.</h3>
08 </body>
09 </html>
```

4.8 web.xml에 추가

web.xml

```
01 <servlet>
02   <servlet-name> controlOne </servlet-name>
03   <servlet-class>controller.ControllerServlet</servlet-class>
04 </servlet>
05 <servlet-mapping>
06   <servlet-name> controlOne </servlet-name>
07   <url-pattern>*.do</url-pattern>
08 </servlet-mapping>
```

4.9 index 페이지 요청방법

<http://localhost:8088/MVCweb/test.do?cmd=index>

페이지 요청의 전달인자로 index를 추가하면 결과를 확인할 수 있다

5 요청 URI 자체를 명령어로 사용하는 MVC

5.1 Command.properties

```
WEB-INF/Command.properties
```

```
01 /mvc/message.do=mvc.MessageProcess
```

5.2 CommandProcess.java

```
src/mvc/Action.java
```

```
01 package mvc;  
02  
03 import javax.servlet.http.HttpServletRequest;  
04 import javax.servlet.http.HttpServletResponse;  
05  
06 //요청 파라미터로 명령어를 전달하는 방식의 슈퍼 인터페이스  
07 public interface CommandProcess {  
08     public String requestPro(HttpServletRequest request,  
09                             HttpServletResponse response) throws Throwable;  
10 }
```

5.3 MessageProcess.java

```
src/mvc/MessageProcess.java
```

```
01 package mvc;  
02  
03 import javax.servlet.http.HttpServletRequest;  
04 import javax.servlet.http.HttpServletResponse;  
05  
06 //Controller로 부터 작업의 처리를 지시받아서 작업을 처리  
07 public class MessageProcess implements CommandProcess {  
08     public String requestPro(HttpServletRequest request, HttpServletResponse response) throws  
09     Throwable {  
10         request.setAttribute("message", "요청 파라미터로 명령어를 전달");  
11  
12         return "/mvc/process.jsp";  
13     }
```

5.4 Controller.java

src/mvc/Controller.java

```

01 package mvc;
02
03 import java.io.*;
04 import java.util.*;
05
06 import javax.servlet.*;
07 import javax.servlet.http.*;
08
09 public class Controller extends HttpServlet {
10     private static final long serialVersionUID = 1L;
11
12     // 명령어와 명령어 처리 클래스를 쌍으로 저장
13     private Map<String, Object> commandMap = new HashMap<String, Object>();
14
15     // 명령어와 처리클래스가 매핑되어 있는 properties 파일을 읽어서 Map객체인
16     commandMap에 저장
17     // 명령어와 처리클래스가 매핑되어 있는 properties 파일은 Command.properties파일
18     @SuppressWarnings("unchecked")
19     public void init(ServletConfig config) throws ServletException {
20         // web.xml에서 propertyConfig에 해당하는 init-param의 값을 읽어옴
21         String props = config.getInitParameter("propertyConfig");
22
23         // 명령어와 처리클래스의 매핑정보를 저장할 Properties객체 생성
24         Properties pr = new Properties();
25         String path = config.getServletContext().getRealPath("/WEB-INF");
26
27         FileInputStream f = null;
28         try {
29             // Command.properties파일의 내용을 읽어옴
30             f = new FileInputStream(new File(path, props));

```

```

31
32     // Command.properties파일의 정보를 Properties객체에 저장
33     pr.load(f);
34 } catch (IOException e) {
35     throw new ServletException(e);
36 } finally {
37     if (f != null)
38         try {
39             fclose();
40         } catch (IOException ex) { }
41 }
42 //Iterator객체는 Enumeration객체를 확장시킨 개념의 객체
43 Iterator<Object> keyIter = pr.keySet().iterator();
44
45 //객체를 하나씩 꺼내서 그 객체명으로 Properties객체에 저장된 객체에 접근
46 while (keyIter.hasNext() ) {
47     String command = (String)keyIter.next();
48     String className = pr.getProperty(command);
49
50     try {
51         //해당 문자열을 클래스로 만든다.
52         Class commandClass = Class.forName(className);
53         Object commandInstance =
54             commandClass.newInstance(); //해당클래스의 객체를 생성
55
56         // Map객체인 commandMap에 객체 저장
57         commandMap.put(command, commandInstance);
58     } catch (ClassNotFoundException e) {
59         throw new ServletException(e);
60     } catch (InstantiationException e) {
61         throw new ServletException(e);
62     } catch (IllegalAccessException e) {
63         throw new ServletException(e);
64     }

```

```
65     }
66 }
67
68 //get방식의 서비스 메소드
69 public void doGet(
70     HttpServletRequest request, HttpServletResponse response)
71     throws ServletException, IOException {
72     requestPro(request, response);
73 }
74
75 // post방식의 서비스 메소드
76 protected void doPost(
77     HttpServletRequest request, HttpServletResponse response)
78     throws ServletException, IOException {
79     requestPro(request, response);
80 }
81
82 //사용자의 요청을 분석해서 해당 작업을 처리
83 private void requestPro(HttpServletRequest request,
84     HttpServletResponse response) throws ServletException, IOException {
85     String view = null;
86     CommandProcess com=null;
87
88     try {
89         String command = request.getRequestURI();
90
91         if (command.indexOf(request.getContextPath()) == 0) {
92             command = command.substring(request.getContextPath().length());
93         }
94
95         com = (CommandProcess)commandMap.get (command);
96         view = com.requestPro(request, response);
97     } catch(Throwable e) {
98         throw new ServletException (e) ;
```

```

99 }
100
101 RequestDispatcher dispatcher = request.getRequestDispatcher(view);
102 dispatcher.forward(request, response);
103 }
104 }
```

5.5 process.jsp

WEB-INF/mvc/process.jsp

```

01 <%@ page contentType="text/html;charset=UTF-8" %>
02 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
03 <html>
04 <head>
05 <title>요청 파라미터로 명령어를 전달하는 예제</title>
06 </head>
07 <body>
08 처리 결과:
09 <cset var="msg" value="${requestScope.message }" />
10 <cout value="${msg }"/>
11 </body>
12 </html>
```

5.6 web.xml에 추가

web.xml

```

01 <servlet>
02   <servlet-name>ControllerURI</servlet-name>
03   <servlet-class>mvc.Controller</servlet-class>
04   <init-param>
05     <param-name>propertyConfig</param-name>
06     <param-value>Command.properties</param-value>
07   </init-param>
08 </servlet>
09 <servlet-mapping>
```

```
10 <servlet-name>ControllerURI</servlet-name>
11 <url-pattern>*.do</url-pattern>
12 </servlet-mapping>
```

5.7 페이지 요청방법

```
http://localhost:8088/MVCweb/mvc/message.do
```

Chapter12. MyBatis 소개 및 설치

1 MyBatis란?

MyBatis는 관계형 데이터베이스의 레코드와 자바 도메인 객체 사이의 매핑을 자동화 해주는 프레임워크로 데이터베이스의 SQL 문을 거의 그대로 사용할 수 있게 해준다. 활용하기 어려운 집계나 조인도 데이터베이스의 SQL 그대로 활용할 수 있게 해준다.

MyBatis 프레임워크는 iBatis 프레임워크를 동일한 개발자들이 재설계하여 새롭게 만든 것이다. 내부적으로는 iBatis라는 패키지 이름이 계속 사용되고 있으며 버전도 이어 받고 있다.

iBatis는 JDK 1.4 이상에서 사용 가능하며 MyBatis는 JDK 1.5 이상을 요구한다. MyBatis 3.2 이상 버전은 JDK 1.6 이상을 요구하고 있다.

1.1 iBatis로부터의 변경점

1.1.1 Package 내부 구조 변경

이름은 변경되었지만 내부적으로는 iBatis와 같은 구조를 사용한다

- iBatis : com.ibatis.*
- MyBatis : org.apache.ibatis.*

1.1.2 SqlMap.xml 내부 구조의 변경

iBatis의 parameterMap이 Deprecated되었으며 기존 부분을 parameterType에 정의하게 되었다

ex)

```
<insert id="insertData" parameterType="DataVo">
    insert into test (user_no, user_name, data, reg_date)
    values ( #{ userNo }, #{ username }, #{ data }, #{ regDate } )
</insert>
```

1.1.3 용어의 변경

- SqlMapConfig -> Configuration
- sqlMap -> mapper

1.1.4 네임스페이스 방식의 변경

sqlMap 별로 약식 명칭을 사용할 수 없게 되었으며 경로를 모두 명시해 주어야 한다. 또한 namespace가 필수 항목이 되었다

- iBatis : <sqlMap namespace="Data">
- MyBatis : <mapper namespace="com.sample.mapper.DataMapper">

2 MyBatis 설치

MyBatis를 사용하기 위해 홈페이지에서 최신버전을 다운 받는다.

현재 최신 버전은 3.4.4 버전이다

<http://blog.mybatis.org/>

사이트를 접속해 Product 탭에서 다운 받을 수 있다

다운받은 압축 파일을 해제하면 mybatis-3.4.4.jar 파일을 확인할 수 있다. jar 파일을 WEB-INF/lib 폴더에 넣어 사용한다. Oracle JDBC를 사용한다면ojdbcXjar파일 또한 WEB-INF/lib 폴더에 넣어서 사용한다.

2.1 MyBatis 설정 파일 (mybatis-config.xml) 생성하기

MyBatis의 설정파일은 getConnection() 메서드의 데이터베이스 연결정보를 대체하는 기능을 제공한다

- Oracle Express Edition에서 SCOTT/TIGER 계정으로 연결을 설정

src/mybatis-config.xml

```

01 <?xml version="1.0" encoding="UTF-8" ?>
02 <!DOCTYPE configuration PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
03   "http://mybatis.org/dtd/mybatis-3-config.dtd">
04 <configuration>
05   <environments default="development">
06     <environment id="development">
07       <transactionManager type="JDBC" />
08       <dataSource type="POOLED">
09         <property name="driver" value="oracle.jdbc.driver.OracleDriver" />
10         <property name="url" value="jdbc:oracle:thin:@localhost:1521:xe" />

```

```

10      <property name="username" value="scott" />
11      <property name="password" value="tiger" />
12    </dataSource>
13  </environment>
14</environments>
15  <mappers>
16    <mapper resource="com/mybatis/user.xml" />
17  </mappers>
18</configuration>

```

2번 Line : 다운받은 mybatis를 압축 해제하면 안에 있는 mybatis-3.4.4.pdf 파일을 확인한다. 이 도움말 파일에서 Instruction 파트의 3페이지에서 복사하여 사용한다.

6번 Line : 트랜잭션 관리자를 설정하는 부분으로 JDBC코드를 대체하는 의미에서 type을 JDBC를 지정한다.

7번 ~ 12번 Line : 데이터베이스 설정

15번 ~ 17번 Line : 매퍼 설정정보를 나타낸다. SQL을 선언해 둔 XML이나 인터페이스 형태의 매퍼 위치를 지정한다

2.2 MyBatis 객체 생성하기 - SqlSessionFactory

만들어진 설정파일을 로드해서 MyBatis 객체를 생성한다

객체 생성 이후 MyBatis를 이용하여 데이터베이스 처리를 하려면 getSqlSessionFactory() 메소드를 이용하여 객체를 생성하고 API를 호출하여 사용하면 된다

```

01 private SqlSessionFactory getSqlSessionFactory() {
02     String resource = "mybatis-config.xml";
03     InputStream inputStream;
04     try {
05         inputStream = org.apache.ibatis.io.Resources.getResourceAsStream(resource);
06     } catch(IOException ioe) {
07         throw new IllegalArgumentException(ioe);
08     }
09     return new SqlSessionFactoryBuilder().build(inputStream);
10 }
11 }

```


Chapter13. myBatis 활용

1 MyBatis 3.x를 이용한 CRUD 예제

mybatis-3.4.4.jar를 WEB-INF/lib 폴더에 넣는다
ojdbc14.jar 파일을 WEB-INF/lib 폴더에 넣는다

1.1 테스트를 위한 테이블 스키마

```
CREATE TABLE "USERTB" (
    "ID" VARCHAR2(12) NOT NULL ENABLE,
    "PASS" VARCHAR2(12) NOT NULL ENABLE,
    "NAME" VARCHAR2(12) NOT NULL ENABLE,
    "AGE" NUMBER(3, 0) DEFAULT 0 NOT NULL ENABLE,
    CONSTRAINT "USER_PK" PRIMARY KEY ("ID") ENABLE
);
```

2 MyBatis 설정파일 작성

src/mybatis-config.xml

```
01 <?xml version="1.0" encoding="UTF-8" ?>
02 <!DOCTYPE configuration PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
03   "http://mybatis.org/dtd/mybatis-3-config.dtd">
04 <configuration>
05   <environments default="development">
06     <environment id="development">
07       <transactionManager type="JDBC" />
08       <dataSource type="POOLED">
09         <property name="driver" value="oracle.jdbc.driver.OracleDriver" />
10         <property name="url" value="jdbc:oracle:thin:@localhost:1521:xe" />
11         <property name="username" value="scott" />
12         <property name="password" value="tiger" />
13       </dataSource>
14     </environment>
15   </environments>
16   <mappers>
17     <mapper resource="com/mybatis/user.xml" />
18   </mappers>
```

18 </configuration>

3 매퍼 파일 작성

com/mybatis/User.xml

```
01 <?xml version="1.0" encoding="UTF-8"?>
02 <!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
03 "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
04 <mapper namespace="com.mybatis.User">
05   <select id="selectAllUsers" resultType="com.mybatis.User">
06     select * from USERTB
07   </select>
08
09   <select id="selectUserById" parameterType="String" resultType="com.mybatis.User">
10     select
11       ID as id,
12       PASS as pass,
13       NAME as name,
14       AGE as age
15     from USERTB
16     where ID = #{id}
17   </select>
18
19   <insert id="insertUser" parameterType="com.mybatis.User">
20     insert into USERTB ( ID, PASS, NAME, AGE )
21     values ( #{id}, #{pass}, #{name}, #{age} )
22   </insert>
23
24   <update id="updateUser" parameterType="com.mybatis.User" >
25     update USERTB
26     set PASS = #{pass},
27       NAME = #{name},
28       AGE = #{age}
29     where ID = #{id}
30   </update>
```

```
30
31     <delete id="deleteUserById" parameterType="String" >
32         delete from USERTB where ID = #{id}
33     </delete>
34 </mapper>
```

4 User VO 작성

com/mybatis/User.java

```
01 package com.mybatis;
02
03 public class User {
04     private String id;
05     private String pass;
06     private String name;
07     private int age;
08
09     @Override
10     public String toString(){
11         return id + ":" + pass + ":" + name + ":" + age;
12     }
13
14     public String getId() {
15         return id;
16     }
17     public void setId(String id) {
18         this.id = id;
19     }
20
21     public String getPass() {
22         return pass;
23     }
24     public void setPass(String pass) {
25         this.pass = pass;
26     }
```

```
27  
28     public String getName() {  
29         return name;  
30     }  
31     public void setName(String name) {  
32         this.name = name;  
33     }  
34  
35     public int getAge() {  
36         return age;  
37     }  
38     public void setAge(int age) {  
39         this.age = age;  
40     }  
41 }
```

5 실행 프로그램 작성

com/mybatis/UserEx.java

```
01 package com.mybatis;  
02  
03 import java.io.BufferedReader;  
04 import java.io.IOException;  
05 import java.io.InputStream;  
06 import java.io.InputStreamReader;  
07  
08 import java.sql.SQLException;  
09  
10 import java.util.List;  
11  
12 import org.apache.ibatis.session.SqlSession;  
13 import org.apache.ibatis.session.SqlSessionFactory;  
14 import org.apache.ibatis.session.SqlSessionFactoryBuilder;  
15  
16 public class UserEx{
```

```
17 private SqlSessionFactory getSqlSessionFactory(){  
18     String resource = "mybatis-config.xml";  
19  
20     InputStream inputStream;  
21     try {  
22         inputStream = org.apache.ibatis.io.Resources.getResourceAsStream(resource);  
23     } catch(IOException ioe) {  
24         throw new IllegalArgumentException(ioe);  
25     }  
26     return new SqlSessionFactoryBuilder().build(inputStream);  
27 }  
28  
29  
30 public List<User> selectAllUsers() throws SQLException {  
31     SqlSession sqlSession = getSqlSessionFactory().openSession();  
32  
33     try {  
34         String statement = "com.mybatis.User.selectAllUsers";  
35  
36         return sqlSession.selectList(statement);  
37     } finally {  
38         sqlSession.close();  
39     }  
40 }  
41  
42 public User selectUserById(String id) {  
43     SqlSession sqlSession = getSqlSessionFactory().openSession();  
44  
45     try {  
46         String statement = "com.mybatis.UserselectUserById";  
47  
48         return (User) sqlSession.selectOne(statement, id);  
49     } finally {  
50         sqlSession.close();  
51     }  
52 }
```

```
51     }
52 }
53
54 public Integer insertUser(User user){
55     SqlSession sqlSession = getSqlSessionFactory().openSession();
56
57     try{
58         String statement = "com.mybatis.User.insertUser";
59         int result = sqlSession.insert(statement, user);
60         if( result>0 ) {
61             sqlSession.commit();
62             return result;
63         }
64         else return 0;
65     } finally {
66         sqlSession.close();
67     }
68 }
69
70 public Integer updateUser(User user) {
71     SqlSession sqlSession = getSqlSessionFactory().openSession();
72
73     try {
74         String statement = "com.mybatis.User.updateUser";
75         int result = sqlSession.update(statement, user);
76         if( result>0 ) {
77             sqlSession.commit();
78         }
79
80         return result;
81     } finally {
82         sqlSession.close();
83     }
84 }
```

```
85 public Integer deleteUser(String id) {  
86     SqlSession sqlSession = getSqlSessionFactory().openSession();  
87  
88     try {  
89         String statement = "com.mybatis.User.deleteUserById";  
90         int result = sqlSession.delete(statement, id);  
91         if( result>0 ) {  
92             sqlSession.commit();  
93         }  
94  
95         return result;  
96     } finally {  
97         sqlSession.close();  
98     }  
99 }  
100  
101 public static void main(String[] ar) throws IOException{  
102     UserEx test = new UserEx();  
103  
104     int menu = -1;  
105     String id = "";  
106     String pass = "";  
107     String name = "";  
108     int age = 0;  
109  
110     User user = null;  
111     BufferedReader br = new BufferedReader(new InputStreamReader(System.in));  
112     while (menu != 0) {  
113         System.out.println("1. 전체목록");  
114         System.out.println("2. 회원검색");  
115         System.out.println("3. 회원등록");  
116         System.out.println("4. 정보수정");  
117         System.out.println("5. 회원탈퇴");  
118         System.out.println("0. 프로그램 종료");
```

```
119
120     System.out.print("메뉴선택 : ");
121     try {
122         menu = Integer.parseInt(br.readLine());
123     } catch (NumberFormatException e) {
124         e.printStackTrace();
125         continue;
126     }
127
128
129     switch (menu) {
130         case 1:
131             List<User> list;
132             try {
133                 list = test.selectAllUsers();
134                 for (User myuser : list) {
135                     System.out.println(myuser.toString());
136                 }
137             } catch (SQLException e) {
138                 e.printStackTrace();
139             }
140
141             break;
142
143         case 2:
144             System.out.print("찾으시는 ID : ");
145             id = br.readLine();
146
147             user = test.selectUserById(id);
148             System.out.println(user.toString());
149             break;
150
151         case 3:
152             System.out.print ("ID = ");
153             id = br.readLine();
```

```
154  
155     System.out.print ("Pass = ");  
156     pass = br.readLine();  
157  
158     System.out.print( "Name = ");  
159     name = br.readLine();  
160  
161     System.out.print ("Age = ");  
162     age = Integer.parseInt(br.readLine());  
163  
164     user = new User();  
165     user.setId(id);  
166     user.setPass(pass);  
167     user.setName(name);  
168     user.setAge(age);  
169  
170  
171     test.insertUser(user);  
172     System.out.println("신규 회원등록 성공");  
173     break;  
174  
175 case 4:  
176     System.out.print("수정할 회원의 ID = ");  
177     id = br.readLine();  
178  
179     System.out.print ("Pass = ");  
180     pass = br.readLine();  
181  
182     System.out.print ("Name = ");  
183     name = br.readLine();  
184  
185     System.out.print ("Age = ");  
186     age = Integer.parseInt(br.readLine());  
187  
188     user = new User();
```

```
189     user.setId(id);
190     user.setPass(pass);
191     user.setName(name);
192     user.setAge(age) ;
193
194     test.updateUser(user);
195     System.out.println(id + "님의 회원정보 수정성공");
196     break;
197
198 case 5:
199     System.out.print("삭제할 회원 ID = ");
200     id = br.readLine();
201
202     test.deleteUser(id);
203     System.out.println(id + "님의 회원정보삭제 성공");
204     break;
205
206 case 0:
207     System.out.println("프로그램을 종료합니다.");
208     break;
209 } // end switch
210 } // end while
211 } // end main
212 }
```

Chapter14. Spring 소개 및 설치

1 Spring이란?

스프링 프레임워크는 엔터프라이즈급 애플리케이션을 만들기 위한 많은 기능을 제공하는 솔루션이다. 필요한 부분만 가져다 사용할 수 있도록 모듈화되어 있다. 자바 애플리케이션 개발을 위한 포괄적인 하부 구조를 제공하는 자바 플랫폼이며 애플리케이션에 집중할 수 있도록 하부 구조를 다룬다.

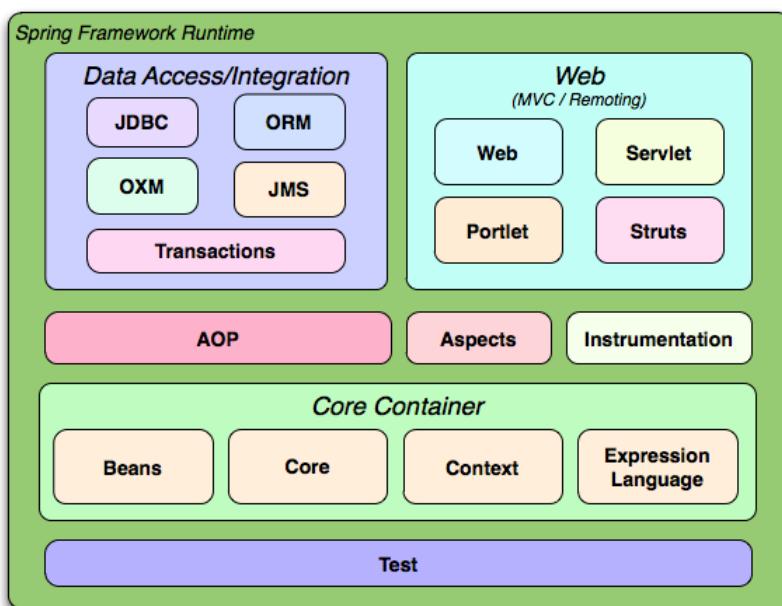
EJB의 단점을 보완한 프레임 워크이며 DI(의존성 주입)와 AOP(관점 지향 프로그래밍)가 지원되는 경량 컨테이너이며 프레임워크이다.

2 Spring의 기능

- 2.1 Spring DI(Dependency Injection) - 의존성 주입
- 2.2 Spring AOP(Aspect Oriented Programming) - 관점 지향 프로그래밍
- 2.3 Spring JDBC
- 2.4 Spring MVC

3 Spring의 구성 모듈

스프링 프레임워크는 약 20개의 모듈로 구성되어있으며, 모듈들은 코어 컨테이너, 데이터 접근/통합 웹, AOP(관점지향 프로그래밍), Instrumentation, 테스트로 그룹을 나눌 수 있다.



3.1 코어 컨테이너

Spring 프레임워크의 근간이 되는 IoC(또는 DI) 기능을 지원하는 영역을 담당하고 있다. BeanFactory를 기반으로 Bean 클래스들을 제어할 수 있는 기능을 지원한다. BeanFactory를 사용하면 프로그래밍 적으로 싱글톤을 구현할 필요가 없고 실제 프로그램 로직에서 의존성에 대한 설정과 명세를 분리할 수 있다.

코어(Core), 빈즈(Beans), 컨텍스트(Context), 표현언어 (Expression Language) 모듈로 이루어졌다.

3.2 데이터 접근/통합 계층

JDBC, ORM, OXM, JMC, 트랜잭션 모듈로 이루어졌다.

JDBC모듈은 JDBC 추상화 계층을 제공한다. JDBC 코딩과 데이터베이스 벤더에 따라 다른 오류코드를 파싱할 필요가 없다.

ORM 모듈은 JPA, JDO , Hibernate, iBatis를 포함하는 인기 있는 객체-관계 매핑 API에 대한 통합 계층을 제공한다. OXM 모듈은 JAXB, Castor, XMLBeans, JiBX, XStream에 대한 객체/XML매핑 구현을 지원하는 계층이다.

자바 메시징 서비스(JMS) 모듈은 메시지를 생산하고 소비하는 기능을 포함한다.

트랜잭션 모듈은 특별한 인터페이스와 모든 POJO (plain old Java objects)의 클래스에 대한 트랜잭션 관리를 지원한다. 트랜잭션 관리는 프로그래밍적으로 하거나 선언적으로 할 수 있다.

3.3 웹 계층

Spring 프레임워크에서 독립적으로 Web UI Layer에 Model-View-Controller를 지원하기 위한 기능이다. 지금까지 Struts, Webwork가 담당했던 기능들을 Spring Web MVC를 이용하여 대체하는 것이 가능하다. 또한 Velocity, Excel, PDF와 같은 다양한 UI 기술들을 사용하기 위한 API를 제공하고 있다.

웹 계층은 웹, 웹-서블릿, 웹-스트러츠, 웹-포틀릿 모듈로 이루어졌다. 웹-서블릿 모듈은 웹 어플리케이션을 위한 스프링의 모델-뷰-컨트롤러 (MVC) 구현을 포함한다. 웹-스트러츠(Web-Struts) 모듈은 스프링 어플리케이션에서 전통적인 스트러츠 웹티어를 통합을 지원하는 클래스를 포함한다. 이 지원은 스프링 3.0에서는 폐기되었다. 웹-포틀릿(Web-Portlet) 모듈은 포틀릿 환경에서 사용되는 MVC 구현과 웹-서블릿 모듈 기능의 미러 기능을 제공한다.

3.4 AOP 계층

Spring 프레임워크에 Aspect Oriented Programming을 지원하는 기능이다. 이 기능은 AOP Alliance 기반 하에서 개발되었다.

4 Spring 설치

4.1 Spring 관련 라이브러리

<http://repo.spring.io/release>

위 사이트로 접속해 org - springframework - spring 경로에서 spring-framework-3.2.9.RELEASE-dist.zip 를 다운 받아 압축 해제한다

4.1.1 Spring 라이브러리의 폴더 구성

- libs - 스프링 프레임워크의 각 모듈별 jar파일과 소스 jar파일을 포함
- docs - API문서 및 레퍼런스 문서 : 도움말
- schema - 각 모듈별 설정 관련 XML 스키마

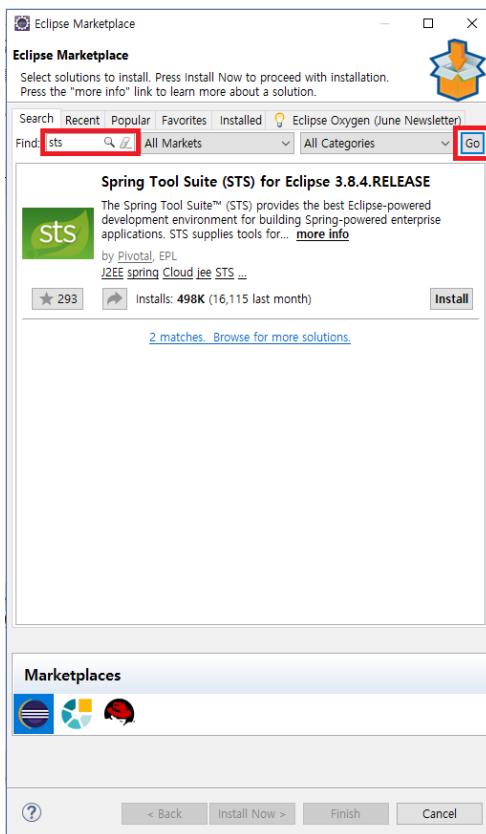
4.1.2 Spring의 모듈 구성

모듈명 (jar파일)	설명
core	DI 기능을 비롯한 프레임워크의 기반을 제공
beans	BeanFactory 인터페이스를 통해 구현
expression	객체에 접근하고 객체를 조작하기 위한 표현 언어를 제공 (JSP 2.1 규약에 명시된 통합 EL을 확장)
context	spring-core와 spring-beans 모듈을 확장해서 국제화, 이벤트 처리, 리소스 로딩, 서블릿 컨테이너를 위한 컨텍스트 생성 등의 기능을 추가로 제공 (ApplicationContext 인터페이스를 통해 구현)
context.support	cache, 메일, 스케줄링, UI의 VeloCity 지원기능을 제공
aop	AOP Alliance에 호환되는 AOP 구현을 제공
aspects	AspectJ와의 통합을 지원
web	파일업로드, IoC 처리 등 웹을 위한 통합 기능을 제공 원격 지원 기능 중 웹 관련 기능을 제공
webmvc	스프링 MVC를 제공한다. JSP, VeloCity에 대한 뷰 연동을 지원
web.struts	스프링과 스트럿츠 연동 기능을 제공
web.portlet	포틀릿 환경에서 사용되는 MVC를 구현
transaction : tx	AOP를 이용한 선언적 트랜잭션 관리 및 코드를 이용한 트랜잭션관리 기능을 제공
jdbс	JDBC프로그래밍을 위한 추상 레이어를 제공 간결한 코드로 JDBC프로그래밍을 할 수 있도록 지원
orm	하이버네이트, JPA, iBatis, JDO등 ORM API를 위한 통합 레이어를 제공

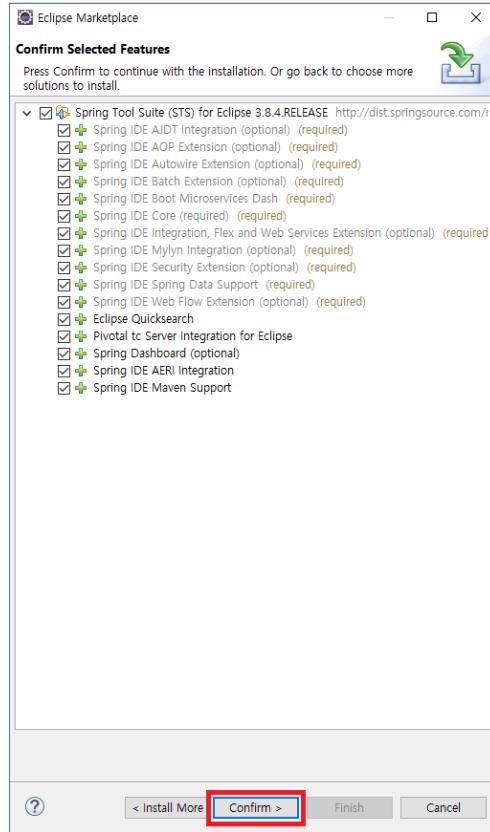
	스프링이 제공하는 트랜잭션 관리와의 연동을 지원
oxm	객체와 XML 사이의 매핑을 처리하기 위한 추상 레이어를 제공 JAXB, Castor, XMLBeans, JiBX, XStream과의 연동을 지원
jms	JMS의 메시지를 생성하고 수신하는 기능을 제공
test	JUnit이나 TestNG를 이용한 스프링 컴포넌트의 테스트를 지원
instrument	instrumentation 지원 클래스를 제공
instrument.tomcat	톰캣 서버를 위한 instrumentation 지원 클래스를 제공

4.2 이클립스에 STS(Spring Tool Suite) 플러그인 설치하기

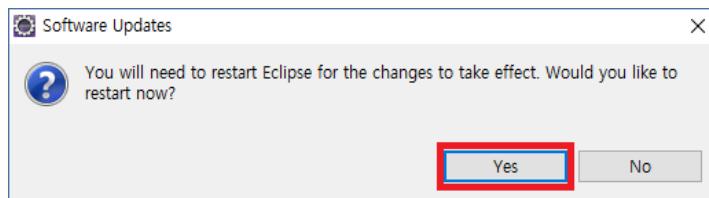
- ① 이클립스를 실행하고 메뉴에서 [Help] - [Eclipse Marketplace]를 클릭
- ② Find : sts 입력하고 [Go] 클릭



- ③ STS(Spring Tool Suite) 이 검색되면 [Install] 클릭
- ④ Features가 모두 선택된 상태에서 [Confirm]



⑤ 설치가 완료되면 Eclipse 재시작



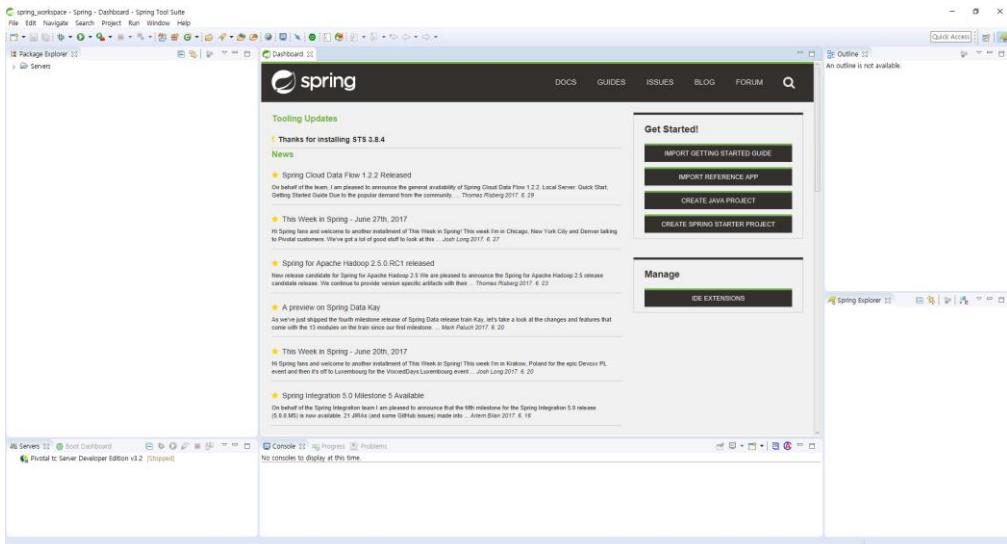
4.3 STS(Spring Tool Suite) 툴 설치

① STS 다운로드 및 설치

<https://spring.io/tools/sts/all> 사이트 접속 후, 원하는 버전 다운

② 다운받은 압축파일을 해제하면 설치 완료

③ sts-3.8.4.RELEASE 폴더의 STS.exe를 실행하면 된다



Chapter15. Spring DI

1 IoC, Inversion of Control - 제어의 역전

IoC란 프로그램의 제어 흐름 구조가 바뀌는 것을 뜻한다.

일반적인 프로그램은 main() 같은 프로그램이 시작되는 지점에서 사용할 객체를 생성하고, 만들어진 객체를 통해 메소드를 호출하는 작업을 통해 프로그램이 수행된다. 객체는 프로그램 흐름을 결정하거나 사용할 객체를 구성하는 작업을 수행하며 프로그램의 흐름에 직접적으로 관여한다. 이러한 방식은 프로그램의 모든 작업을 사용하는 쪽에서 제어하는 구조이다.

이와 달리 IoC는 제어 흐름의 개념을 거꾸로 뒤집은 것이다. 객체는 자신이 어떻게 만들어지고 어디서 사용되는지 알 수 없으며 사용할 객체를 생성하거나 선택하지 않는다. 객체의 제어 권한을 객체 자신이 아닌 다른 대상에게 위임하는 것이다. 객체의 생성부터 생명주기의 관리까지 모든 객체에 대한 제어 권한을 다른 대상에게 위임한 것을 제어권의 역전(IoC)이라는 개념이다.

- 작업을 수행하는 쪽에서 객체를 생성하는 기준의 제어 흐름 개념을 뒤집은 것이다
- 사용할 객체를 생성하거나 선택하지 않는다
- 모든 객체는 제어 권한을 위임받는 특별한 객체에 의해서 만들어 지고 사용된다
- 객체는 자신이 어떻게 생성되고 어떻게 사용되는지 알 수 없다

2 DI, Dependency Injection - 의존성 주입

객체 간의 의존관계를 객체 자신이 아닌 외부 조립기가 수행해주는 개념

각 계층 사이, 각 클래스 사이에서 필요한 의존 관계를 컨테이너에 자동적으로 연결시켜주는 것으로 Bean설정 정보를 바탕으로 컨테이너가 자동적으로 연결한다. 불필요한 의존 관계를 없애거나 줄일 수 있으며, 단위테스트 수행이 수월해진다.

설정 파일이나 어노테이션을 이용하여 객체 간의 의존 관계를 설정할 수 있다. DI 패턴에는 생성자 방식과, 메소드 방식이 존재한다.

2.1 직접 객체를 생성하며 의존적인 코드

dependency/Person.java

```
01 package dependency;  
02  
03 public class Person {  
04     public static void main(String[] args) {  
05         Car car = new Car();  
06     }  
07 }
```

```
07     System.out.println(car.getTire( ));  
08 }  
09 }
```

dependency/Car.java

```
01 package dependency;  
02  
03 public class Car {  
04     Tire tire;  
05  
06     public Car( ) {  
07         tire = new GoldTire( );  
08     }  
09  
10    public String getTire( ) {  
11        return tire.getProduct( ) + "를 장착하였습니다";  
12    }  
13 }
```

dependency/Tire.java

```
01 package dependency;  
02  
03 public interface Tire {  
04     String getProduct( );  
05 }
```

dependency/GoldTire.java

```
01 package dependency;  
02  
03 public class GoldTire implements Tire {  
04     @Override  
05     public String getProduct( ) {  
06         return "골드 타이어";  
07     }
```

```
08 }
```

```
dependency/SilverTire.java
```

```
01 package dependency;  
02  
03 public class SilverTire implements Tire {  
04     @Override  
05     public String getProduct() {  
06         return "실버 타이어";  
07     }  
08 }
```

자동차는 타이어에 의존적이며, 사람은 자동차에 의존적이다

(new를 통해 객체를 생성하여 사용하면서 해당 클래스의 객체에 의존적이 된다고 볼 수 있다)

2.2 생성자를 통한 의존성 주입

Car 객체 코드 내에 Tire 객체를 생성해야하는 의존적인 부분을 없애고 Person 객체에서 Car 객체를 생성할 때 Tire를 선택하도록 해줄 수 있게 됨으로써 코드의 유연함이 생긴다.

```
constructorDI/Person.java
```

```
01 package constructorDI;  
02  
03 public class Person {  
04     public static void main(String[] args) {  
05  
06         Tire gTire = new GoldTire();  
07         Tire sTire = new SilverTire();  
08  
09         Car car1 = new Car(gTire);  
10         System.out.println(car1.getTire());  
11  
12         Car car2 = new Car(sTire);  
13         System.out.println(car2.getTire());  
14     }
```

```
15 }
```

constructorDI/Car.java

```
01 package constructorDI;  
02  
03 public class Car {  
04     Tire tire;  
05  
06     // DI를 위한 constructor  
07     public Car(Tire tire) {  
08         this.tire = tire;  
09     }  
10  
11     public String getTire() {  
12         return tire.getProduct() + "를 장착하였습니다";  
13     }  
14 }
```

constructorDI/Tire.java

```
01 package constructorDI;  
02  
03 public interface Tire {  
04     String getProduct();  
05 }
```

constructorDI/GoldTire.java

```
01 package constructorDI;  
02  
03 public class GoldTire implements Tire {  
04     @Override  
05     public String getProduct() {  
06         return "골드 타이어";  
07     }  
08 }
```

constructorDI/SilverTire.java

```
01 package constructorDI;  
02  
03 public class SilverTire implements Tire {  
04     @Override  
05     public String getProduct() {  
06         return "실버 타이어";  
07     }  
08 }
```

2.3 setter를 통한 의존성 주입

setterDI/Person.java

```
01 package setterDI;  
02  
03 public class Person {  
04     public static void main(String[] args) {  
05  
06         Tire gTire = new GoldTire();  
07         Tire sTire = new SilverTire();  
08  
09         Car car = new Car();  
10  
11         car.setTire(gTire);  
12         System.out.println(car.getTire());  
13  
14         car.setTire(sTire);  
15         System.out.println(car.getTire());  
16     }  
17 }
```

setterDI/Car.java

```
01 package setterDI;
```

```
02  
03 public class Car {  
04     Tire tire;  
05  
06     // DI를 위한 setter  
07     public void setTire(Tire tire) {  
08         this.tire = tire;  
09     }  
10  
11     public String getTire() {  
12         return tire.getProduct() + "를 장착하였습니다";  
13     }  
14 }
```

setterDI/Tire.java

```
01 package setterDI;  
02  
03 public interface Tire {  
04     String getProduct();  
05 }
```

setterDI/GoldTire.java

```
01 package setterDI;  
02  
03 public class GoldTire implements Tire {  
04     @Override  
05     public String getProduct() {  
06         return "골드 타이어";  
07     }  
08 }
```

setterDI/SilverTire.java

```
01 package setterDI;  
02
```

```
03 public class SilverTire implements Tire {  
04     @Override  
05     public String getProduct() {  
06         return "실버 타이어";  
07     }  
08 }
```

3 스프링에서의 의존성 주입

3.1 XML 파일 이용

springDIxml/Person.java

```
01 package springDIxml;  
02  
03 import org.springframework.context.ApplicationContext;  
04 import org.springframework.context.support.FileSystemXmlApplicationContext;  
05  
06 public class Person {  
07     public static void main(String[] args) {  
08  
09         ApplicationContext context =  
10             new FileSystemXmlApplicationContext("/src/main/java/springDIxml/di.xml");  
11  
12         Tire gTire = (Tire) context.getBean("gTire");  
13         Car car1 = (Car) context.getBean("car1");  
14  
15         car1.setTire(gTire);  
16         System.out.println( car1.getInfo() );  
17  
18  
19         Tire sTire = (Tire) context.getBean("sTire");  
20         Car car2 = (Car) context.getBean("car2");  
21  
22         car2.setTire(sTire);  
23         System.out.println( car2.getInfo() );
```

```
24  
25     }  
26 }
```

springDlxml/Car.java

```
01 package springDlxml;  
02  
03 public class Car {  
04     Tire tire;  
05  
06     public void setTire(Tire tire) {  
07         this.tire = tire;  
08     }  
09  
10    public Tire getTire() {  
11        return tire;  
12    }  
13  
14    public String getInfo() {  
15        return tire.getProduct() + "를 장착하였습니다";  
16    }  
17 }
```

springDlxml/Tire.java

```
01 package springDlxml;  
02  
03 public interface Tire {  
04     String getProduct();  
05 }
```

dependency/GoldTire.java

```
01 package springDlxml;  
02  
03 public class GoldTire implements Tire {
```

```

04  @Override
05  public String getProduct() {
06      return "골드 타이어";
07  }
08 }
```

springDIxml/SilverTire.java

```

01 package springDIxml;
02
03 public class SilverTire implements Tire {
04     @Override
05     public String getProduct() {
06         return "실버 타이어";
07     }
08 }
```

- 의존성 주입을 위한 XML 파일 생성

STS 의 New - Other - Spring - Spring Bean Configuration File 을 선택하여 쉽게 생성 가능

springDIxml/dixml

```

01 <?xml version="1.0" encoding="UTF-8"?>
02 <beans xmlns="http://www.springframework.org/schema/beans"
03   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
04   xsi:schemaLocation="http://www.springframework.org/schema/beans
05   http://www.springframework.org/schema/beans/spring-beans.xsd">
06
07     <bean id="gTire" class="springDIxml.GoldTire"/>
08     <bean id="sTire" class="springDIxml.SilverTire"/>
09
10    <bean id="car1" class="springDIxml.Car" />
11    <bean id="car2" class="springDIxml.Car" />
12
13  </beans>
```

3.2 XML 파일 내의 속성을 이용

springDIxmlProperty/Person.java

```
01 package springDIxmlProperty;  
02  
03 import org.springframework.context.ApplicationContext;  
04 import org.springframework.context.support.FileSystemXmlApplicationContext;  
05  
06 public class Person {  
07     public static void main(String[] args) {  
08  
09         ApplicationContext context =  
10             new FileSystemXmlApplicationContext("/src/main/java/springDIxmlProperty/di.xml");  
11  
12         Car car1 = (Car) context.getBean("car1");  
13         System.out.println( car1.getInfo() );  
14  
15         Car car2 = (Car) context.getBean("car2");  
16         System.out.println( car2.getInfo() );  
17  
18     }  
19 }
```

Tire 를 설정하는 항목이 제거됨

springDIxmlProperty/Car.java

```
01 package springDIxmlProperty;  
02  
03 public class Car {  
04     Tire tire;  
05  
06     public void setTire(Tire tire) {  
07         this.tire = tire;  
08     }  
09  
10     public Tire getTire() {  
11         return tire;
```

```
12 }
13
14     public String getInfo() {
15         return tire.getProduct() + "를 장착하였습니다";
16     }
17 }
```

springDIxmlProperty/Tire.java

```
01 package springDIxmlProperty;
02
03 public interface Tire {
04     String getProduct();
05 }
```

springDIxmlProperty/GoldTire.java

```
01 package springDIxmlProperty;
02
03 public class GoldTire implements Tire {
04     @Override
05     public String getProduct() {
06         return "골드 타이어";
07     }
08 }
```

springDIxmlProperty/SilverTire.java

```
01 package springDIxmlProperty;
02
03 public class SilverTire implements Tire {
04     @Override
05     public String getProduct() {
06         return "실버 타이어";
07     }
08 }
```

- 의존성 주입을 위한 XML 파일 생성

bean 항목의 property를 이용하여 Tire에 대한 의존성을 주입

springDIxmlProperty/di.xml

```
01 <?xml version="1.0" encoding="UTF-8"?>
02 <beans xmlns="http://www.springframework.org/schema/beans"
03   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
04   xsi:schemaLocation="http://www.springframework.org/schema/beans
05   http://www.springframework.org/schema/beans/spring-beans.xsd">
06
07   <bean id="gTire" class="springDIxmlProperty.GoldTire"/>
08   <bean id="sTire" class="springDIxmlProperty.SilverTire"/>
09
10   <bean id="car1" class="springDIxmlProperty.Car">
11     <property name="tire" ref="gTire" />
12   </bean>
13   <bean id="car2" class="springDIxmlProperty.Car">
14     <property name="tire" ref="sTire" />
15   </bean>
16
17 </beans>
```

3.3 @Autowired 를 이용

springDIautowired/Person.java

```
01 package springDIautowired;
02
03 import org.springframework.context.ApplicationContext;
04 import org.springframework.context.support.FileSystemXmlApplicationContext;
05
06 public class Person {
07
08   public static void main(String[] args) {
09
10     ApplicationContext context =
11       new FileSystemXmlApplicationContext("/src/main/java/springDIautowired/di.xml");
```

```
12  
13     Car car1 = (Car) context.getBean("car1");  
14     System.out.println( car1.tire.getProduct() );  
15  
16     Car car2 = (Car) context.getBean("car2");  
17     System.out.println( car2.tire.getProduct() );  
18  
19 }  
20 }
```

springDIautowired/Car.java

```
01 package springDIautowired;  
02  
03 import org.springframework.beans.factory.annotation.Autowired;  
04  
05 public class Car {  
06  
07     @Autowired  
08     Tire tire;  
09  
10     public String getInfo() {  
11         return tire.getProduct() + "를 장착하였습니다";  
12     }  
13 }
```

springDIautowired/Tire.java

```
01 package springDIautowired;  
02  
03 public interface Tire {  
04     String getProduct();  
05 }
```

springDIautowired/GoldTire.java

```
01 package springDIautowired;
```

```

02
03 public class GoldTire implements Tire {
04     @Override
05     public String getProduct() {
06         return "골드 타이어";
07     }
08 }
```

springDIautowired/SilverTire.java

```

01 package springDIautowired;
02
03 public class SilverTire implements Tire {
04     @Override
05     public String getProduct() {
06         return "실버 타이어";
07     }
08 }
```

- xml파일을 생성한 후 Namespaces에서 context 항목을 추가함

springDIautowired/di.xml

```

01 <?xml version="1.0" encoding="UTF-8"?>
02 <beans xmlns="http://www.springframework.org/schema/beans"
03   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
04   xmlns:context="http://www.springframework.org/schema/context"
05   xsi:schemaLocation="http://www.springframework.org/schema/beans
06   http://www.springframework.org/schema/beans/spring-beans.xsd
07   http://www.springframework.org/schema/context
08   http://www.springframework.org/schema/context/spring-context-3.1.xsd">
09
10    <!-- 추가 -->
11    <context:annotation-config />
12
13    <bean id="gtire" class="springDIautowired.GoldTire"/>
14    <bean id="tire" class="springDIautowired.SilverTire"/>
```

```
15      <bean id="car1" class="springDIautowired.Car" />
16      <bean id="car2" class="springDIautowired.Car" />
17
18
19  </beans>
```

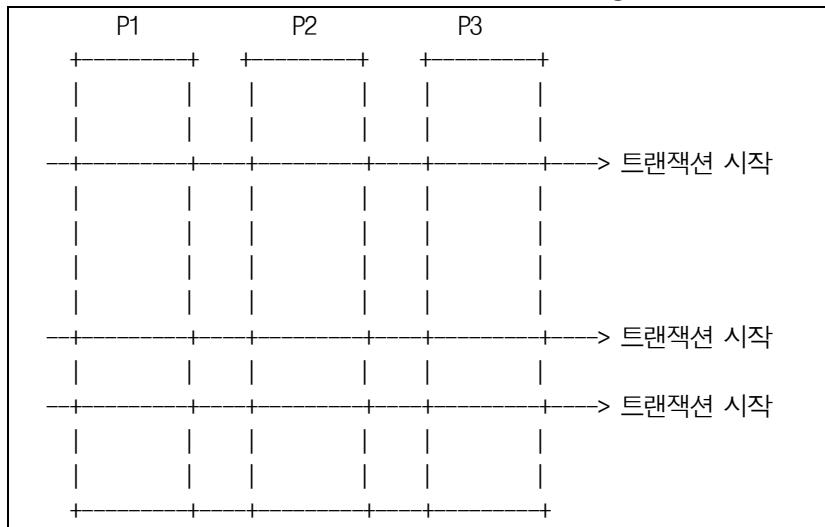
Chapter16. Spring AOP

1 AOP, Aspect-Oriented Programming

관점 지향 프로그래밍

로직(code)에 대한 관점을 이해하고 적용하는 방법론이다

프로그램에서는 특정 코드가 반복적으로 나오게 된다. 아래의 그림처럼 프로그램들을 횡단하는 공통적인 코드가 나타나게 되는데 이를 횡단적 관심사(Cross-cutting Concern)이라고 한다.



프로그래머는 이러한 부분을 추출하여 클래스로 만들고 반복적으로 호출하여 사용할 수 있도록 공통 모듈로 작성하게 된다. 하지만 이러한 공통 모듈을 호출하는 코드가 반드시 프로그램 존재하게 되는데 이러한 호출 부분까지 꺼내어 비즈니스 로직을 처리할 때 호출 코드까지 필요없게 만들어 주는 것이 AOP이다.

공통 코드를 작성한 후 실행되어야 할 위치만 설정을 해주면 자동으로 코드가 끼워넣어진 것처럼 동작하게 된다.

이 때 공통적으로 뽑아낸 코드를 Advice라고 하며 코드를 끼워 넣을 곳을 Joinpoint, 조인포인트를 모아둔 것을 Pointcut이라고 한다. 어드바이스를 조인포인트에 실제로 끼워넣는 작업을 Weaving이라고 한다. 스프링에서는 어드바이스와 포인트 커스터터를 묶어 Advisor라고 한다.

Advice는 조인포인트에서 실행되는 순서에 따라 타입을

- Before Advice : Joinpoint 앞에서 실행되는 Advice
- Around Advice : Joinpoint 앞과 뒤에서 실행되는 Advice
- After Advice : Joinpoint 호출이 리턴되기 직전에 실행되는 Advice
- After Returning Advice : Joinpoint 메소드 호출이 정상적으로 종료된 후에 실행되는 Advice
- After Throwing Advice : 예외가 발생했을 때 실행되는 Advice

로 구분한다.

2 Advice 예제

2.1 pom.xml에 의존성 추가

```
<dependency>
    <groupId>org.aspectj</groupId>
    <artifactId>aspectjweaver</artifactId>
    <version>1.6.11</version>
</dependency>

<dependency>
    <groupId>cglib</groupId>
    <artifactId>cglib</artifactId>
    <version>2.2</version>
</dependency>
```

2.2 Advice 클래스 작성

com/spring/aop/adviceTest.java

```
01 package com.spring.aop;
02
03 import org.aspectj.lang.JoinPoint;
04 import org.aspectj.lang.ProceedingJoinPoint;
05 import org.aspectj.lang.annotation.After;
06 import org.aspectj.lang.annotation.AfterReturning;
07 import org.aspectj.lang.annotation.AfterThrowing;
08 import org.aspectj.lang.annotation.Around;
09 import org.aspectj.lang.annotation.Aspect;
10 import org.aspectj.lang.annotation.Before;
11 import org.aspectj.lang.annotation.Pointcut;
12
13 @Aspect
14 public class AdviceTest {
```

```

16 // 공통으로 사용될 포인트컷 지정
17 // com.spring.aop.controller 패키지 안의 Controller 로 끝나는 클래스의 모든 메소드에 적용됩니다.
18 @Pointcut("execution(* com.spring.aop.*Controller.*(..))")
19 public void commonPointcut() { }
20
21
22 // Before Advice 입니다. 위에서 정의한 공통 포인터 컷을 사용합니다.
23 @Before("commonPointcut()")
24 public void beforeMethod(JoinPoint jp) throws Exception {
25     System.out.println("beforeMethod() called.....");
26
27     // 인자들을 얻기
28     Object arg[] = jp.getArgs();
29
30     // 인자의 갯수 출력
31     System.out.println("args length : " + arg.length);
32
33     // 첫 번째 인자의 클래스 명 출력
34     System.out.println("arg0 name : " + arg[0].getClass().getName());
35
36     // 호출될 메소드 명 출력
37     System.out.println(jp.getSignature().getName());
38 }
39
40 // After Advice
41 @After("commonPointcut()")
42 public void afterMethod(JoinPoint jp) throws Exception {
43     System.out.println("afterMethod() called.....");
44 }
45
46 // After Returning Advice 입니다.
47 // 반환값을 받을 수 있음
48 @AfterReturning(pointcut="commonPointcut()", returning="returnString")
49 public void afterReturningMethod(JoinPoint jp, String returnString) throws Exception {

```

```

50
51     System.out.println("afterReturningMethod() called.....");
52     // 호출된 메소드 반환값 출력
53     System.out.println("afterReturningMethod() returnString : " + returnString);
54 }
55
56 // Around Advice
57 // 포인트컷을 직접 지정
58 @Around("execution(* com.spring.aop.*Controller.*(..))")
59 public Object aroundMethod(ProceedingJoinPoint pjp) throws Throwable {
60
61     System.out.println("aroundMethod() before called.....");
62
63     Object result = pjp.proceed();
64     System.out.println("aroundMethod() after called.....");
65
66     return result;
67 }
68
69 // 예외가 발생했을때 Advice
70 @AfterThrowing(pointcut="commonPointcut()", throwing="exception")
71 public void afterThrowingMethod(JoinPoint jp, Exception exception) throws Exception {
72     System.out.println("afterThrowingMethod() called.....");
73
74     // 발생한 예외의 메세지를 출력합니다.
75     System.out.println(exception.getMessage());
76 }
77
78 }

```

2.3 Advice 등록

WEB-INF/spring-appServlet-servlet-context.xml

01	<?xml version="1.0" encoding="UTF-8"?>
02	<beans:beans xmlns="http://www.springframework.org/schema/mvc"

```

03    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
04    xmlns:beans="http://www.springframework.org/schema/beans"
05    xmlns:context="http://www.springframework.org/schema/context"
06    xmlns:aop="http://www.springframework.org/schema/aop"
07    xsi:schemaLocation="http://www.springframework.org/schema/mvc
08        http://www.springframework.org/schema/mvc/spring-mvc.xsd
09            http://www.springframework.org/schema/beans
10        http://www.springframework.org/schema/beans/spring-beans.xsd
11            http://www.springframework.org/schema/context
12        http://www.springframework.org/schema/context/spring-context.xsd
13            http://www.springframework.org/schema/aop
14        http://www.springframework.org/schema/aop/spring-aop-3.1.xsd">
15
16        <!-- DispatcherServlet Context: defines this servlet's request-processing infrastructure -->
17        <!-- Enables the Spring MVC @Controller programming model -->
18        <annotation-driven />
19
20        <aop:aspectj-autoproxy />
21        <beans:bean id="AdviceTest" class="com.spring.aop.AdviceTest" />
22
23        <!-- Handles HTTP GET requests for /resources/** by efficiently serving up static
24 resources in the ${webappRoot}/resources directory -->
25        <resources mapping="/resources/**" location="/resources/" />
26
27        <!-- Resolves views selected for rendering by @Controllers to .jsp resources in the
28 /WEB-INF/views directory -->
29        <beans:bean
30        class="org.springframework.web.servlet.view.InternalResourceViewResolver">
31            <beans:property name="prefix" value="/WEB-INF/views/" />
32            <beans:property name="suffix" value=".jsp" />
33        </beans:bean>
34
35        <context:component-scan base-package="com.spring.aop" />
36

```

2.4 컨트롤러 작성

com/spring/aop/TestController.java

```

01 package com.spring.aop;
02
03 import org.springframework.stereotype.Controller;
04 import org.springframework.ui.Model;
05 import org.springframework.web.bind.annotation.RequestMapping;
06 import org.springframework.web.bind.annotation.RequestMethod;
07
08 @Controller
09 public class TestController {
10
11     @RequestMapping(value = "/aop.do", method = RequestMethod.GET)
12     public String aop(Model model) {
13         System.out.println("call aop.do");
14
15         return "aopTest";
16     }
17 }
```

2.5 뷰 작성

WEB-INF/views/aopTest.jsp

```

01 <%@ page language="java" contentType="text/html; charset=UTF-8"
02     pageEncoding="UTF-8"%>
03 <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
04
05 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
06 "http://www.w3.org/TR/html4/loose.dtd">
07 <html>
08 <head>
09 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
```

```
10 <title>AOP</title>
11 </head>
12 <h1>AOP COMPLETE!</h1>
13 </body>
14 </html>
```

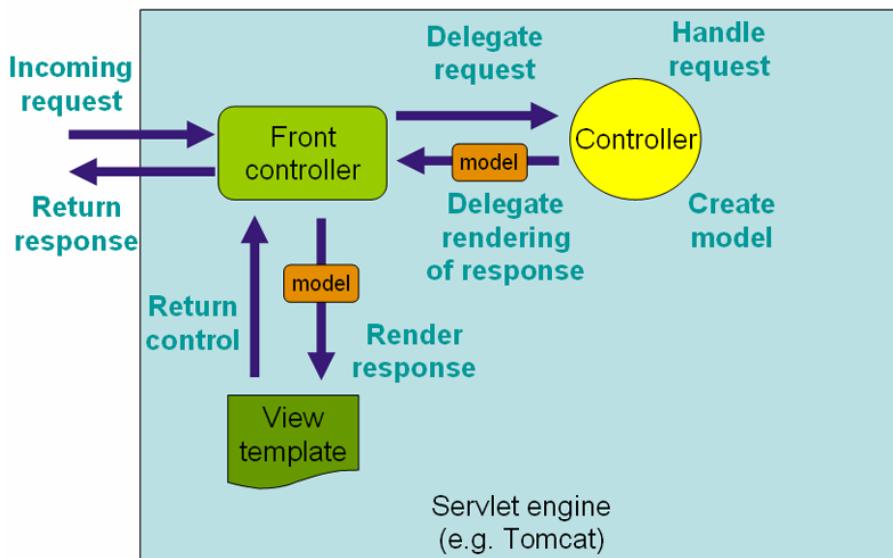
Chapter17. Spring MVC

1 스프링 MVC

스프링은 비즈니스 로직을 표현하는 도메인 모델(Model)과 프레젠테이션을 표현하는 View를 분리하고 사이에 컨트롤러를 배치하도록 설계한 디자인 패턴인 MVC 패턴을 지원한다

스프링 MVC란 스프링이 제공하는 웹 어플리케이션 구축 전용 MVC 프레임워크를 말한다. 스프링 MVC를 이용해 Model, View, Controller 사이의 의존 관계를 DI 컨테이너에서 관리하고 통일되고 유연한 웹 어플리케이션을 구축할 수 있다.

2 스프링 MVC의 클라이언트 요청 처리 과정



- ① 웹 브라우저의 요청(Request)이 DispatcherServlet 인스턴스로 송신된다.
- ② 요청을 받은 DispatcherServlet 인스턴스는 어플리케이션 안에서 공통 처리를 실행한 후 requestURL 고유의 처리를 실행하는 Request 컨트롤러(Controller 인스턴스)를 호출한다.
- ③ RequestURL과 Request컨트롤러의 맵핑을 관리하는 HandlerMapping 인스턴스와 DispatcherServlet 인스턴스를 참조하여 HandlerMapping 인스턴스로부터 반환된 Controller로 처리를 전달한다.
- ④ 처리가 전달된 Controller는 필요한 비즈니스 로직을 호출하여 처리 결과(모델)와 이동할 View 정보를 DispatcherServlet에 반환한다.
(모델과 View 는 스프링MVC가 제공하는 ModelAndView 인스턴스로 취급된다)
- ⑤ DispatcherServlet 인스턴스는 View의 실체를 ViewResolver 인스턴스에 확인한다.

(Controller 인스턴스로부터 반환된 View는 논리 정보이기 때문)

- ⑥ DispatcherServlet은 ViewResolver 인스턴스에 의해 해결된 View 인스턴스에 대해 모델을 렌더링하여 처리 결과를 브라우저에 표시한다.

3 스프링 MVC의 구성요소

구성 요소	설 명
DispatcherServlet	클라이언트의 요청을 전달받음 요청에 맞는 컨트롤러가 리턴한 결과값을 View에 전달하여 알맞은 응답을 생성
HandlerMapping	클라이언트의 요청 URL을 어떤 컨트롤러가 처리할지 결정
Controller	클라이언트의 요청을 처리한 뒤, 결과를 DispatcherServlet에게 리턴
ModelAndView	컨트롤러가 처리한 결과 정보 및 뷰 선택에 필요한 정보를 담음
ViewResolver	컨트롤러의 처리 결과를 생성할 뷰를 결정
View	컨트롤러의 처리 결과 화면을 생성, JSP나 Velocity 템플릿 파일 등을 뷰로 사용

4 스프링 MVC를 이용한 개발

- ① 클라이언트의 요청을 받을 DispatcherServlet을 web.xml 파일에 설정
- ② 클라이언트의 요청을 처리할 컨트롤러 작성
- ③ ViewResolver 설정 (컨트롤러가 전달한 값을 이용해서 응답 화면을 생성할 뷰를 결정)
- ④ JSP나 Velocity 등을 이용하여 뷰 영역의 코드를 작성

5 스프링 MVC를 이용하여 Hello World 출력하기

스프링 MVC를 이용하여 Hello World를 출력하는 웹 어플리케이션을 작성해 본다

5.1 Dynamic Web Project 를 생성한다

5.2 lib를 추가한다

- commons-logging-1.1.1.jar
- spring-beans-3.2.9.RELEASE.jar
- spring-context-3.2.9.RELEASE.jar
- spring-context-support-3.2.9.RELEASE.jar

- spring-core-3.2.9.RELEASE.jar
- spring-expression-3.2.9.RELEASE.jar
- spring-web-3.2.9.RELEASE.jar
- spring-webmvc-3.2.9.RELEASE.jar

5.3 Dispatcher Servlet 설정

web.xml 파일에 DispatcherServlet을 설정한다

```
<servlet>
    <servlet-name>dispatcher</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
</servlet>
```

5.4 공통 어플리케이션 컨텍스트 설정

web.xml 파일에 어플리케이션 컨텍스트를 설정한다

```
<servlet-mapping>
    <servlet-name>dispatcher</servlet-name>
    <url-pattern>*.do</url-pattern>
</servlet-mapping>
```

*.do로 들어오는 클라이언트의 요청을 DispatcherServlet이 처리하도록 설정

DispatcherServlet은 WEB-INF/ 폴더 내에 있는 [서블릿이름]-servlet.xml 파일을 스프링 설정파일로 사용한다

WEB-INF/web.xml

01	<?xml version="1.0" encoding="UTF-8"?>
02	<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
03	xmlns="http://xmlns.jcp.org/xml/ns/javaee"
04	xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-
05	app_3_1.xsd"
06	id="WebApp_ID" version="3.1">

```

07 <display-name>MVC_HELLO_1</display-name>
08 <welcome-file-list>
09   <welcome-file>index.html</welcome-file>
10   <welcome-file>index.htm</welcome-file>
11   <welcome-file>index.jsp</welcome-file>
12   <welcome-file>default.html</welcome-file>
13   <welcome-file>default.htm</welcome-file>
14   <welcome-file>default.jsp</welcome-file>
15 </welcome-file-list>
16
17 <servlet>
18   <servlet-name>dispatcher</servlet-name>
19   <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
20 </servlet>
21
22 <servlet-mapping>
23   <servlet-name>dispatcher</servlet-name>
24   <url-pattern>*.do</url-pattern>
25 </servlet-mapping>
26 </web-app>

```

5.5 컨트롤러 구현

컨트롤러를 구현하기 위해서는 @Controller 어노테이션을 클래스에 적용한다

src/com/mvc/controller/HelloController.java

```

01 package com.mvc.controller;
02
03 import java.util.Calendar;
04
05 import org.springframework.stereotype.Controller;
06 import org.springframework.web.bind.annotation.RequestMapping;
07 import org.springframework.web.servlet.ModelAndView;
08
09 @Controller

```

```

10 public class HelloController {
11
12     @RequestMapping("/hello.do")
13     public ModelAndView hello() {
14         ModelAndView mav = new ModelAndView();
15
16         mav.setViewName("hello");
17         mav.addObject("greeting", getGreeting());
18
19         return mav;
20     }
21
22     private String getGreeting() {
23         int hour = Calendar.getInstance().get(Calendar.HOUR_OF_DAY);
24
25         if (hour >= 6 && hour <= 10) {
26             return "좋은 아침입니다.";
27         } else if (hour >= 12 && hour <= 15) {
28             return "점심 식사는 하셨나요?";
29         } else if (hour >= 18 && hour <= 22) {
30             return "좋은 밤 되세요";
31         }
32
33         return "안녕하세요";
34     }
35 }

```

@RequestMapping 어노테이션을 통해 클라이언트의 요청을 처리할 메소드를 지정하고 있다
 "/hello.do"로 처리할 경로를 지정하면서 http://localhost:port/mvc/hello.do 요청을 HelloController 클래스의 hello() 메소드가 처리하게 된다

5.6 스프링 설정 파일 추가

5.6.1 컨트롤러 등록

DispatcherServlet이 스프링 컨테이너에서 컨트롤러를 객체를 검색할 수 있도록 스프링 설정파일에 컨트롤러 빈을 등록한다

```
<beans:bean id="helloController" class="com.mvc.controller.HelloController" />
```

5.6.2 ViewResolver 등록

DispatcherServlet이 스프링 컨테이너에서 컨트롤러를 객체를 검색할 수 있도록 스프링 설정파일에 컨트롤러 빈을 등록한다

```
<beans:bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
    <beans:property name="prefix" value="/WEB-INF/view/" />
    <beans:property name="suffix" value=".jsp" />
</beans:bean>
```

ViewResolver 가 "/view/뷰이름.jsp" 형식의 뷰 JSP를 사용함을 알리는 설정

WEB-INF/dispatcher-servlet.xml

```
01  <?xml version="1.0" encoding="UTF-8"?>
02  <beans:beans xmlns="http://www.springframework.org/schema/mvc"
03      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
04      xmlns:beans="http://www.springframework.org/schema/beans"
05      xmlns:context="http://www.springframework.org/schema/context"
06      xsi:schemaLocation="http://www.springframework.org/schema/mvc
07          http://www.springframework.org/schema/mvc/spring-mvc.xsd
08          http://www.springframework.org/schema/beans
09          http://www.springframework.org/schema/beans/spring-beans.xsd
10          http://www.springframework.org/schema/context
11          http://www.springframework.org/schema/context/spring-context.xsd">
12
13  <!-- DispatcherServlet Context: defines this servlet's request-processing infrastructure -->
14
15  <!-- Enables the Spring MVC @Controller programming model -->
16  <annotation-driven />
17
18  <!-- Resolves views selected for rendering by @Controllers to .jsp resources in the /WEB-
```

```

19  INF/views directory -->
20    <beans:bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
21      <beans:property name="prefix" value="/WEB-INF/view/" />
22      <beans:property name="suffix" value=".jsp" />
23    </beans:bean>
24
25    <beans:bean id="helloController" class="com.mvc.controller.HelloController" />
26
27    <context:component-scan base-package="com.mvc" />
28
29  </beans:beans>
30

```

5.7 뷰 작성

WEB-INF/view/hello.jsp

```

01  <%@ page language="java" contentType="text/html; charset=EUC-KR"
02    pageEncoding="EUC-KR"%>
03  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
04  "http://www.w3.org/TR/html4/loose.dtd">
05  <html>
06  <head>
07  <meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
08  <title>Insert title here</title>
09  </head>
10  <body>
11
12  인사말 : <strong>${greeting }</strong>
13
14  </body>
15  </html>

```

HelloController 컨트롤러의 addObject()메소드를 통해 전달한 greeting 값을 뷰 JSP에서 사용할 수 있게 된다

6 DispatcherServlet 설정과 ApplicationContext의 관계

DispatcherServlet은 클라이언트의 요청을 중앙에서 처리하는 스프링 MVC의 핵심 구성요소이다. web.xml 파일에 한 개 이상의 DispatcherServlet을 설정할 수 있으며 각 DispatcherServlet은 한 개의 WebApplicationContext를 갖게 된다. 또한 각 DispatcherServlet이 공유하는 빈을 설정할 수 도 있다.

6.1 DispatcherServlet 설정

DispatcherServlet은 기본적으로 웹 어플리케이션의 /WEB-INF/[서블릿이름]-servlet.xml 파일로부터 스프링 설정 정보를 읽어옴

```
<servlet>
    <servlet-name>dispatcher</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <init-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>
            /WEB-INF/beans.xml
            /WEB-INF/main.xml
        </param-value>
    </init-param>
</servlet>
```

한 개 이상의 설정파일을 사용해야 하는 경우나 기본설정이 아닌 다른 이름을 사용하고 싶다면 contextConfigLocation 초기화 파라미터를 설정하면 된다. contextConfigLocation 초기화 파라미터는 설정 파일 목록을 값으로 갖는데 이때 각 설정파일은 콤마(","), 공백문자(" "), 탭("\t"), 줄바꿈("\n"), 세미콜론(";")을 이용하여 구분한다.

6.2 웹 어플리케이션을 위한 ApplicationContext 설정

DispatcherServlet은 그 자체가 서블릿이기 때문에 한 개 이상의 DispatcherServlet을 설정하는 것 이 가능

```
<servlet>
    <servlet-name>dispatcher</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
```

```

<init-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>
        /WEB-INF/beans.xml
        /WEB-INF/main.xml
    </param-value>
</init-param>
</servlet>

```

두 DispatcherServlet은 각각 별도의 WebApplicationContext를 생성하게 된다

front.xml에서는 rest.xml에 설정된 빈 객체를 사용할 수 없다는 점을 주의

만약 공통의 빈의 필요로 하는 경우 ContextLoaderListener를 사용하여 공통으로 사용될 빈을 설정 할 수 있다.

```

<context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>
        /WEB-INF/service.xml
        /WEB-INF/persistence.xml
    </param-value>
</context-param>
<listener>
    <listener-class>
        org.springframework.web.context.ContextLoaderListener
    </listener-class>
</listener>
<servlet>
    <servlet-name> front </servlet-name>
    <servlet-class>
        org.springframework.web.servlet.DispatcherServlet
    </servlet-class>
</servlet>
<servlet>
    <servlet-name> rest </servlet-name>

```

```
</servlet-name>
<servlet-class>
    org.springframework.web.servlet.DispatcherServlet
</servlet-class>
</servlet>
```

ContextLoaderListener를 ServletListener로 등록하고 contextConfigLocation 컨텍스트 파라미터를 이용하여 공통으로 사용될 빈 정보를 담고 있는 설정 파일 목록을 지정

```
<context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>
        /WEB-INF/service.xml
        /WEB-INF/persistence.xml
    </param-value>
</context-param>
<listener>
    <listener-class>
        org.springframework.web.context.ContextLoaderListener
    </listener-class>
</listener>
<servlet>
    <servlet-name> front </servlet-name>
    <servlet-class>
        org.springframework.web.servlet.DispatcherServlet
    </servlet-class>
</servlet>
<servlet>
    <servlet-name>re st
    </servlet-name>
    <servlet-class>
        org.springframework.web.servlet.DispatcherServlet
    </servlet-class>
</servlet>
```

ContextLoaderListener가 생성하는 WebApplicationContext는 웹 어플리케이션에서 루트 컨텍스트가 되며 DispatcherServlet이 생성하는 WebApplicationContext는 루트 컨텍스트를 부모로 사용하는 자식 컨텍스트가 된다.

ContextLoaderListener는 contextConfigLocation 컨텍스트 파라미터를 명시하지 않으면 /WEB-INF/applicationContext.xml을 설정파일로 사용한다. 만약 클래스 패스에 위치한 파일로부터 설정 정보를 읽어오고 싶다면 "classpath:" 접두어를 사용해서 명시한다

7 캐릭터 인코딩 처리를 위한 필터 설정

요청 파라미터의 캐릭터 인코딩이 ISO-8859-1이 아닌 경우 setCharacterEncoding() 메소드를 사용해서 인코딩을 해주어야 한다

```
response.setCharacterEncoding("UTF-8");
```

web.xml파일에 CharacterEncodingFilter 클래스를 설정함으로써 모든 컨트롤러에 한 번에 인코딩을 적용시킬 수 있다

```
<filter>
    <filter-name>encodingFilter</filter-name>
    <filter-class>
        org.springframework.web.filter.CharacterEncodingFilter
    </filter-class>
    <init-param>
        <param-name>encoding</param-name>
        <param-value>
            UTF-8
        </param-value>
    </init-param>
</filter>
<filter-mapping>
    <filter-name>encodingFilter</filter-name>
    <url-pattern>/* </url-pattern>
</filter-mapping>
```

8 컨트롤러 구현

스프링 3.0 부터는 @Controller 어노테이션을 이용해서 컨트롤러를 클래스를 구현하도록 권장

8.1 @Controller 어노테이션과 @RequestMapping 어노테이션

컨트롤러 클래스를 구현하려면 @Controller 어노테이션과 @RequestMapping 어노테이션을 이용

- 컨트롤러 클래스에 @Controller 어노테이션을 적용
- 클라이언트의 요청을 처리할 메소드에 @RequestMapping 어노테이션을 적용
- 설정 파일에 컨트롤러 클래스를 빈으로 등록

```
@Controller  
public class HelloController {  
    @RequestMapping( "/hello.do")  
    public String hello(){  
        return "HelloSpring";  
    }  
}
```

@RequestMapping 어노테이션은 해당 메서드에서 처리할 URI를 값으로 갖는다.

실제로 처리할 URI는 web.xml에서 설정한 DispatcherServlet에 설정한 매팅과 스프링 설정 파일의 전체 경로 이용 여부 설정에 따라 달라진다. hello() 메소드의 리턴 타입이 String인데 이 경우 메서드의 리턴 값으로 컨트롤러의 처리 결과를 보여줄 뷰 이름으로 사용하게 된다. 컨트롤러 클래스를 구현하면 DispatcherServlet이 사용하는 스프링 설정 파일에 해당 컨트롤러 클래스를 등록해 준다

```
<bean id="helloController" class="com.lsjspring.controller.HelloController" />
```

8.2 컨트롤러 메서드의 HTTP 전송방식 한정

하나의 요청 URR에 대해 GET 요청과 POST 요청을 한 개의 컨트롤러에서 처리해야 할 경우, @RequestMapping 어노테이션의 method 속성을 이용해서 처리할 HTTP 메서드를 제한할 수 있다.

```
@Controller  
@RequestMapping("/board/newBoard.do")
```

```

public class NewBoardController {
    @RequestMapping( method = RequestMethod.GET)
    public String form() {
        return "board/newBoardForm";
    }

    @RequestMapping(value="/board/newBoard.do", method = RequestMethod.POST)
    public String submit () {
        return "board/newBoardSubmitted";
    }
}

```

클라이언트 요청을 처리하는 두 메소드가 동일한 URI를 처리할 경우 @RequestMapping 어노테이션을 클래스에 적용해서 해당 클래스가 처리할 기본 URI를 지정할 수 있다

@RequestMapping 어노테이션에 method 속성을 설정하지 않을 경우 모든 HTTP 메소드를 처리하게 된다

9 뷰 지정

컨트롤러 처리 메소드는 처리 결과를 보여줄 뷰 이름이나 View 객체를 리턴하고 DispatcherServlet은 뷰 이름이나 View 객체를 이용해서 뷰를 생성한다

9.1 뷰 이름 명시적 지정

뷰이름을 명시적으로 지정하려면 ModelAndView나 String을 리턴해야 한다.

```

@RequestMapping("/index.do")
public ModelAndView index(){
    ModelAndView mav = new ModelAndView( "index");
    //...
    return mav;
}

```

ModelAndView를 리턴할 경우 ModelAndView 클래스의 생성자나 setViewName() 메서드를 이용해서 뷰 이름을 지정할 수 있다

```
ModelAndView mav = new ModelAndView( );
mav.setViewName("search/game");
```

String 타입을 리턴할 경우 문자열 값이 뷰 이름으로 사용된다

```
@RequestMapping("/help/main.do")
public String helpMain(ModelMap model) {
    //...
    return "help/main";
}
```

9.2 뷰 이름 자동 지정

- 리턴 타입이 Model이나 Map인 경우
- 리턴 타입이 void이면서 ServletResponse나 HttpServletResponse 타입의 파라미터가 없는 경우

RequestToViewNameTranslator를 이용해서 URL로부터 뷰 이름을 결정한다. 스프링 설정 파일에 RequestToViewNameTranslator 빈이 존재하지 않을 경우 기본적으로 DefaultRequestToViewNameTranslator를 사용한다

- 요청 URL로부터 맨 앞의 슬래시와 확장자를 제외한 나머지 부분을 뷰 이름으로 사용

```
@RequestMapping("/search/game.do")
public Map<String, Object> search( ) {
    HashMap<String, Object> model = new HashMap<String, Object>();
    //...
    return model;
}
```

/search/game.do == > search/game

9.3 리다이렉트 뷰

뷰 이름에 "redirect:" 접두어를 붙이면 지정한 페이지로 리다이렉트 된다.

```
redirect : /bbs/list - 현재 서블릿 컨텍스트에 대한 상대적인 경로로 리다이렉트
redirect : http://host//bbs/list - 지정한 절대 URL로 리다이렉트
```

```
ex) mav.setViewName("redirect : /error.do"); // 에러페이지로 이동
```

Chapter18. Spring MVC 구현

1 파일 업로드 처리

파일 업로드가 필요한 경우 HTML 폼의 enctype 속성을 "multipart/form-data"로 설정한다.

```
<form method="post" enctype="multipart/form-data">
```

인코딩 타입이 Multipart인 경우 파라미터나 업로드 한 파일을 구하려면 전송 데이터를 알맞게 처리해 주어야 한다. 이를 위해서 스프링은 Multipart 지원 기능을 제공하고 있기 때문에 추가적인 처리 없이 Multipart 형식으로 전송된 파라미터와 파일정보를 쉽게 구할 수 있다.

1.1 MultipartResolver 설정

Multipart 지원 기능을 사용하려면 먼저 MultipartResolver를 스프링 설정파일에 등록해 주어야 한다. MultipartResolver는 Multipart 형식으로 데이터가 전송된 경우 스프링 MVC가 사용할 수 있도록 변환해 준다.

스프링이 기본적으로 제공하는 MultipartResolver는 CommonsMultipartResolver이다. Commons MultipartResolver는 Commons FileUpload API를 이용한다.

```
<bean id="multipartResolver">
    class="org.springframework.web.multipart.commons.CommonsMultipartResolver">
</bean>
```

DispatcherServlet은 이름이 "multipartResolver"인 빈을 사용하기 때문에 다른 이름을 지정할 경우 MultipartResolver로 사용되지 않는다.

- CommonsMultipartResolver 클래스의 프로퍼티

프로퍼티	타입	설명
maxUploadSize	long	최대 업로드 가능한 바이트 크기 기본 값은 -1이다 (제한 없음)
maxInMemorySize	int	디스크에 임시 파일을 생성하기 전에 메모리에 보관할 수 있는 최대 바이트 크기 기본 값은 10240 바이트
defaultEncoding	String	요청을 파싱할 때 사용할 캐릭터 인코딩 지정하지 않을 경우 HttpServletRequest.setCharacterEncoding() 메소드로 지정한 캐릭터 셋을 사용

	아무 값도 없을 때는 ISO-8859-1을 사용
--	----------------------------

1.2 첫번째 방법 : @RequestParam 어노테이션을 이용한 업로드 파일의 접근

업로드 한 파일을 전달받는 첫 번째 방법은 @RequestParam 어노테이션이 적용된 MultipartFile 타입의 파라미터를 사용하는 것이다.

- FORM HTML

```
<form action="submitReport1.do" method="post" enctype="multipart/form-data">  
학번:<input type="text" name="studentNumber" /><br>  
리포트파일:<input type="file" name="report" /><br>  
<input type="submit" />  
</form>
```

위 HTML 코드에서 파일은 report 파라미터를 통해서 전달되며 아래의 Controller에서와 같이 @RequestParam 어노테이션과 MultipartFile 타입의 파라미터를 이용해서 업로드 파일 데이터를 전달 받을 수 있다.

- Controller

```
package com.mvc.file.controller;  
// 중략 ...  
import org.springframework.web.bind.annotation.RequestParam;  
import org.springframework.web.multipart.MultipartFile;  
  
@Controller  
public class ReportSubmissionController {  
  
    @RequestMapping( value = "/report/submission.do", method = RequestMethod.GET )  
    public String form() {  
        return "report/submissionForm";  
    }  
  
    @RequestMapping( value = "/report/submitReport1.do", method = RequestMethod.POST )  
    public String submitReport1(  
        @RequestParam("studentNumber") String studentNumber,
```

```

@RequestParam("report") MultipartFile report) {
    //
    // MultipartFile이 제공하는 메소드를 이용해서 업로드 데이터 접근
    printInfo(studentNumber, report);

    return "report/submissionComplete";
}
// 중략 ...
}

```

1.3 두 번째 방법 : MultipartHttpServletRequest를 이용한 업로드파일 접근

MultipartHttpServletRequest 인터페이스는 스프링이 제공하는 인터페이스로서 Multipart 요청이 들어올 때 내부적으로 원본 HttpServletRequest 대신 사용되는 인터페이스이다

- Controller

```

// 중략...
import org.springframework.web.multipart.MultipartHttpServletRequest;
@Controller
public class ReportSubmissionController {
    @RequestMapping( value = "/report/submitReport2.do", method = RequestMethod.POST)
    public String submitReport2(MultipartHttpServletRequest request) {

        String studentNumber = request.getParameter("studentNumber");
        MultipartFile report = request.getFile ("report");
        printInfo(studentNumber, report);
        return "report/submissionComplete";
    }
    // 중략 ...
}

```

MultipartHttpServletRequest는 인터페이스이며 실제로 어떤 메서드도 선언되어 있지 않다

MultipartHttpServletRequest는 HttpServletRequest를 상속받기 때문에 웹 요청 정보를 구하기 위한 getParameter()와 같은 메서드를 사용할 수 있으며 MultipartRequest가 제공하는 메서드도 사용할 수 있다

- MultipartRequest 인터페이스의 파일 관련 주요 메서드

메소드	설명
Iterator<String> getFileNames()	업로드 된 파일의 이름 목록을 제공하는 Iterator를 구함
MultipartFile getFile(String name)	파라미터 이름이 name인 업로드 파일 정보를 구함
List<MultipartFile> getFiles(String name)	파라미터 이름이 name인 업로드 파일 정보 목록을 구함
Map <String, MultipartFile> getFile()	파라미터 이름을 키로 파라미터에 해당하는 파일 정보를 값으로 하는 Map을 구함

1.4 세 번째 방법 : 커맨드 객체를 통한 업로드 파일 접근

커맨드 클래스에 파라미터와 동일한 이름의 MultipartFile 타입 프로퍼티를 추가해주기만 하면 커맨드 객체를 이용해 업로드 한 파일을 전달 받을 수 있게 된다.

- Command 클래스

```
import org.springframework.web.multipart.MultipartFile;

public class ReportCommand {
    private String studentNumber;
    private MultipartFile report;

    String getStudentNumber() {
        return studentNumber;
    }

    void setStudentNumber(String studentNumber) {
        this.studentNumber = studentNumber;
    }

    MultipartFile getReport() {
        return report;
    }

    void setReport(MultipartFile report) {
```

```
        this.report = report;
    }
}
```

위와 같이 MultipartFile 타입의 프로퍼티를 커맨드 클래스에 추가해 주었다면 @RequestMapping 메서드의 커맨드 객체로 사용함으로써 업로드 파일 정보를 커맨드 객체를 통해서 전달받을 수 있다.

- Controller

```
@Controller
public class ReportSubmissionController {
    // 중략 ...
    @RequestMapping( value = "/report/submitReport3.do", method = RequestMethod.POST)
    public String submitReport3 (ReportCommand report Command) {
        printInfo(reportCommand.getStudentNumber(), reportCommand.getReport());
        return "report/submissionComplete";

    // 중 락 ...
    }
}
```

2 파일 업로드 처리

2.1 이클립스에서 프로젝트를 생성하고 라이브러리를 추가한다

- commons-logging-1.1.1.jar
- spring-beans-3.2.9.RELEASE.jar
- spring-beans-3.2.9.RELEASE.jar
- spring-context-support-3.2.9.RELEASE.jar
- spring-core-3.2.9.RELEASE.jar
- spring-expression-3.2.9.RELEASE.jar
- spring-web-3.2.9.RELEASE.jar
- spring-webmvc-3.2.9.RELEASE.jar

2.2 web.xml에 DispatcherServlet을 등록

WEB-INF/web.xml

```
01 <?xml version="1.0" encoding="UTF-8"?>
02 <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
03   xmlns="http://xmlns.jcp.org/xml/ns/javaee" xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
04   http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd" id="WebApp_ID" version="3.1">
05   <display-name>FileUp</display-name>
06   <welcome-file-list>
07     <welcome-file>index.html</welcome-file>
08     <welcome-file>index.htm</welcome-file>
09     <welcome-file>index.jsp</welcome-file>
10     <welcome-file>default.html</welcome-file>
11     <welcome-file>default.htm</welcome-file>
12     <welcome-file>default.jsp</welcome-file>
13   </welcome-file-list>
14
15   <filter>
16     <filter-name>encodingFilter</filter-name>
17     <filter-class>
18       org.springframework.web.filter.CharacterEncodingFilter
19     </filter-class>
20
21     <init-param>
22       <param-name>encoding</param-name>
23       <param-value>UTF-8</param-value>
24     </init-param>
25   </filter>
26
27   <filter-mapping>
28     <filter-name>encodingFilter</filter-name>
29     <url-pattern>/*</url-pattern>
30   </filter-mapping>
31
32   <servlet>
```

```

33 <servlet-name>spring</servlet-name>
34 <servlet-class>
35   org.springframework.web.servlet.DispatcherServlet
36 </servlet-class>
37 </servlet>
38
39 <servlet-mapping>
40   <servlet-name>spring</servlet-name>
41   <url-pattern>*.do</url-pattern>
42 </servlet-mapping>
43
44 </web-app>

```

2.3 스프링 설정파일을 작성

WEB-INF/spring-servlet.xml

```

01 <?xml version="1.0" encoding="UTF-8"?>
02 <beans xmlns="http://www.springframework.org/schema/beans"
03   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
04   xmlns:context="http://www.springframework.org/schema/context"
05   xmlns:p="http://www.springframework.org/schema/p"
06   xsi:schemaLocation="http://www.springframework.org/schema/beans
07     http://www.springframework.org/schema/beans/spring-beans.xsd
08     http://www.springframework.org/schema/context
09     http://www.springframework.org/schema/context/spring-context-3.1.xsd">
10
11 <bean id="multipartResolver"
12   class="org.springframework.web.multipart.commons.CommonsMultipartResolver">
13 </bean>
14
15 <bean class="com.mvc.controller.ReportController" />
16
17 <bean id="viewResolver"
18   class="org.springframework.web.servlet.view.InternalResourceViewResolver">
19

```

```

20 <property name="prefix" value="/WEB-INF/" />
21 <property name="suffix" value=".jsp" />
22 </bean>
23 </beans>

```

2.4 리포트 제출 폼 작성

WEB-INF/report/reportForm.jsp

```

01 <?xml version="1.0" encoding="UTF-8" ?>
02 <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
03
04 <%@taglib uri="http://www.springframework.org/tags" prefix="spring" %>
05 <%@taglib uri="http://www.springframework.org/tags/form" prefix="form" %>
06
07 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
08 "http://www.w3.org/TR/xhtml/DTD/xhtml1-transitional.dtd">
09
10 <html xmlns="http://www.w3.org/1999/xhtml">
11 <head>
12 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
13 <title>리포트 제출</title>
14 </head>
15 <body>
16
17 <h3>@RequestParam 사용</h3>
18 <form action="submitReport1.do" method="post"
19 enctype="multipart/form-data">
20   학번:<input type="text" name="studentNumber" /><br/>
21   리포트파일:<input type="file" name="report" /> <br/>
22   <input type="submit" />
23 </form>
24
25 <h3>MultipartHttpServletRequest 사용</h3>
26 <form action="submitReport2.do" method="post" enctype="multipart/form-data">
27   학번:<input type="text" name="studentNumber" /><br/>

```

```

28     리포트파일:<input type="file" name="report" /> <br/>
29     <input type="submit" />
30   </form>
31
32   <h3>커맨드 객체 사용</h3>
33   <form action="submitReport3.do" method="post" enctype="multipart/form-data">
34     학번:<input type="text" name="studentNumber" /> <br/>
35     리포트파일:<input type="file" name="report" /><br/>
36     <input type="submit" />
37   </form>
38 </body>
39 </html>

```

2.5 컨트롤러 작성

src/com/mvc/controller/ReportController.java

```

01 package com.mvc.controller;
02
03 import org.springframework.stereotype.Controller;
04 import org.springframework.web.bind.annotation.RequestMapping;
05 import org.springframework.web.bind.annotation.RequestMethod;
06 import org.springframework.web.bind.annotation.RequestParam;
07 import org.springframework.web.multipart.MultipartFile;
08 import org.springframework.web.multipart.MultipartHttpServletRequest;
09
10 @Controller
11 public class ReportController {
12
13     @RequestMapping(value="/report/reportForm.do", method=RequestMethod.GET)
14     public String form() {
15         return "report/reportForm";
16     }
17
18     @RequestMapping( value="/report/submitReport1.do", method=RequestMethod.POST)
19     public String submitReport1 (

```

```

20     @RequestParam("studentNumber") String studentNumber,
21     @RequestParam("report") MultipartFile report) {
22         printInfo(studentNumber, report);
23
24         return "report/reportComplete";
25     }
26
27     private void printInfo(String studentNumber, MultipartFile report) {
28         System.out.println(studentNumber + "가 업로드 한 파일: " + report.getOriginalFilename() );
29     }
30
31     @RequestMapping( value="/report/submitReport2.do", method=RequestMethod.POST)
32     public String submitReport2(MultipartHttpServletRequest request) {
33         String studentNumber = request.getParameter( "studentNumber");
34         MultipartFile report = request.getFile("report");
35
36         printInfo(studentNumber, report);
37         return "report/reportComplete";
38     }
39
40     @RequestMapping( value="/report/submitReport3.do", method=RequestMethod.POST)
41     public String submitReport3(ReportCommand reportCommand) {
42         printInfo(reportCommand.getStudentNumber(), reportCommand.getReport());
43
44         return "report/reportComplete";
45     }
46 }
```

2.6 커맨드 클래스 작성

src/com/mvc/controller/ReportController.java

```

01 package com.mvc.controller;
02
03 import org.springframework.web.multipart.MultipartFile;
04
```

```

05 public class ReportCommand {
06     private String studentNumber;
07     private MultipartFile report;
08
09     public String getStudentNumber() {
10         return studentNumber;
11     }
12     public void setStudentNumber(String studentNumber) {
13         this.studentNumber = studentNumber;
14     }
15
16     public MultipartFile getReport() {
17         return report;
18     }
19     public void setReport(MultipartFile report) {
20         this.report = report;
21     }
22 }
```

2.7 전송 완료 화면 작성

WEB-INF/report/reportComplete.jsp

```

01 <?xml version="1.0" encoding="UTF-8"?>
02
03 <%@ page language="java" contentType="text/html; charset=UTF-8"
04     pageEncoding="UTF-8"%>
05
06 <%@taglib uri="http://www.springframework.org/tags" prefix="spring" %>
07 <%@taglib uri="http://www.springframework.org/tags/form" prefix="form" %>
08
09 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
10 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
11
12 <html xmlns="http://www.w3.org/1999/xhtml">
13 <head>
```

```
14 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
15 <title>리포트 제출 완료</title>
16 </head>
17 <body>리포트 제출 완료
18 </body>
19 </html>
```

2.8 확인

<http://localhost:port/projectName/report/reportForm.do> 로 접속하여 확인할 수 있다

3 스프링 MVC 게시판 구현

댓글 기능을 가지고 있는 게시판 구현

3.1 데이터베이스 테이블 구현

3.1.1 게시글 저장 테이블

```
CREATE TABLE "BOARDDC" (
    "NUM" NUMBER(7 , 0) NOT NULL ENABLE,
    "WRITER" VARCHAR2(12) NOT NULL ENABLE,
    "EMAIL" VARCHAR2(30) NOT NULL ENABLE,
    "SUBJECT" VARCHAR2(50) NOT NULL ENABLE,
    "PASS" VARCHAR2(10),
    "READCOUNT" NUMBER(5 , 0) DEFAULT 0 NOT NULL ENABLE,
    "REGDATE" DATE DEFAULT SYSDATE NOT NULL ENABLE,
    "CONTENT" VARCHAR2(4000) NOT NULL ENABLE,
    CONSTRAINT "BOARDDC_PK" PRIMARY KEY ("NUM") ENABLE
);
```

3.1.2 게시글 시퀀스

```
CREATE SEQUENCE "BOARDDC_SEQ" START WITH 1 INCREMENT BY 1
NOCACHE NOCYCLE NOORDER;
```

3.1.3 댓글 테이블

```
CREATE TABLE "BOARD_DC_COMMENT" (
    "COMMENT_ID" NUMBER(7,0) NOT NULL ENABLE,
    "NUM" NUMBER(7,0) NOT NULL ENABLE,
    "COMMENT_NAME" VARCHAR2(20),
    "COMMENTS" VARCHAR2(4000),
    "REGDATE" DATE DEFAULT SYSDATE NOT NULL ENABLE,
    CONSTRAINT "BOARD_DC_COMMENT_PK"
        PRIMARY KEY ("COMMENT_ID") ENABLE
);
```

3.1.4 댓글 시퀀스

```
CREATE SEQUENCE "BOARD_DC_COMMENT_SEQ" START WITH 1 INCREMENT BY 1
NOCACHE NOCYCLE NOORDER;
```

3.2 추가 Library

3.2.1 AspectJ 관련

- aspectjrt-1.6.10.jar
- aspectjweaver-1.6.11.jar
- aopalliance-1.0.jar

3.2.2 CGLIB 관련

- cglib-2.2.jar

3.2.3 Apache FileUpload API 관련

- commons-fileupload-1.2.2.jar
- commons-io-2.0.1.jar

3.2.4 Common Logging API 관련

- commons-logging-1.1.1.jar

3.2.5 Database 연동 관련

- mybatis-3.4.4.jar
- mybatis-spring-1.2.1.jar
- ojdbc14.jar

3.2.6 JSTL 관련

- jstl-1.2.jar

3.2.7 스프링 관련

- spring-aop-3.2.9.RELEASE.jar
- spring-beans-3.2.9.RELEASE.jar
- spring-context-3.2.9.RELEASE.jar
- spring-context-support-3.2.9.RELEASE.jar
- spring-core-3.2.9.RELEASE.jar
- spring-expression-3.2.9.RELEASE.jar
- spring-jdbc-3.2.9.RELEASE.jar
- spring-orm-3.2.9.RELEASE.jar
- spring-tx-3.2.9.RELEASE.jar
- spring-web-3.2.9.RELEASE.jar
- spring-webmvc-3.2.9.RELEASE.jar

3.3 기본 환경 설정

3.3.1 WebApplication 환경 설정 - web.xml

WEB-INF/web.xml

01	<?xml version="1.0" encoding="UTF-8"?>
02	<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
03	xmlns="http://xmlns.jcp.org/xml/ns/javaee" xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
04	http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd" id="WebApp_ID" version="3.1">
05	<display-name>board_comment</display-name>
06	<welcome-file-list>
07	<welcome-file>index.html</welcome-file>
08	<welcome-file>index.htm</welcome-file>

```
09    <welcome-file>index.jsp</welcome-file>
10    <welcome-file>default.html</welcome-file>
11    <welcome-file>default.htm</welcome-file>
12    <welcome-file>default.jsp</welcome-file>
13  </welcome-file-list>
14
15
16  <filter>
17    <filter-name>CharacterEncodingFilter</filter-name>
18    <filter-class>
19      org.springframework.web.filter.CharacterEncodingFilter
20    </filter-class>
21
22    <init-param>
23      <param-name>encoding</param-name>
24      <param-value>UTF-8 </param-value>
25    </init-param>
26    <init-param>
27      <param-name>forceEncoding</param-name>
28      <param-value>true</param-value>
29    </init-param>
30  </filter>
31
32  <filter-mapping>
33    <filter-name>CharacterEncodingFilter</filter-name>
34    <url-pattern>/*</url-pattern>
35  </filter-mapping>
36
37  <context-param>
38    <param-name>contextConfigLocation</param-name>
39    <param-value>
40      /WEB-INFcontexts/applicationContext.xml
41      /WEB-INFcontexts/commentContext.xml
42      /WEB-INFcontexts/boardContext.xml
```

```

43   </param-value>
44 </context-param>
45
46 <listener>
47   <listener-class>
48     org.springframework.web.context.ContextLoaderListener
49   </listener-class>
50 </listener>
51
52 <servlet>
53   <servlet-name>spring</servlet-name>
54   <servlet-class>
55     org.springframework.web.servlet.DispatcherServlet
56   </servlet-class>
57   <load-on-startup>1</load-on-startup>
58 </servlet>
59
60 <servlet-mapping>
61   <servlet-name>spring</servlet-name>
62   <url-pattern>*.do</url-pattern>
63 </servlet-mapping>
64
65 </web-app>

```

3.3.2 스프링 설정파일 - applicationContext.xml

WEB-INF/contextes/applicationContext.xml : bean, p, context, tx

```

01 <?xml version="1.0" encoding="UTF-8"?>
02 <beans xmlns="http://www.springframework.org/schema/beans"
03   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
04   xmlns:context="http://www.springframework.org/schema/context"
05   xmlns:tx="http://www.springframework.org/schema/tx"
06   xmlns:p="http://www.springframework.org/schema/p"
07   xsi:schemaLocation="http://www.springframework.org/schema/beans
08   http://www.springframework.org/schema/beans/spring-beans.xsd"

```

```

09      http://www.springframework.org/schema/context
10     http://www.springframework.org/schema/context/spring-context-3.2.xsd
11     http://www.springframework.org/schema/tx
12     http://www.springframework.org/schema/tx/spring-tx-3.2.xsd">
13
14     <!-- Data Source -->
15     <bean id="dataSource"
16       class="org.springframework.jdbc.datasource.DriverManagerDataSource">
17       <!-- JDBC 드라이버 클래스명 설정 -->
18       <property name="driverClassName">
19         <value>oracle.jdbc.driver.OracleDriver</value>
20       </property>
21
22       <!-- JDBC 접속 문자열 설정 -->
23       <property name="url">
24         <value>jdbc:oracle:thin:@localhost:1521:xe</value>
25       </property>
26       <property name="username">
27         <value>scott</value>
28       </property>
29       <property name="password">
30         <value>tiger</value>
31       </property>
32     </bean>
33
34     <!-- Transaction 관리자 설정 -->
35     <bean id="transactionManager"
36       class="org.springframework.jdbc.datasource.DataSourceTransactionManager"
37       p:dataSource-ref="dataSource" />
38
39     <!-- myBatis 사용을 위한 설정 -->
40     <bean id="sqlSessionFactoryBean" class="org.mybatis.spring.SqlSessionFactoryBean">
41       <property name="dataSource" ref="dataSource" />
42       <property name="configLocation" value="classpath:/mybatis-config.xml" />

```

```

43   </bean>
44   <bean id="sqlSessionTemplate" class="org.mybatis.spring.SqlSessionTemplate">
45     <constructor-arg name="sqlSessionFactory" ref="sqlSessionFactoryBean" />
46   </bean>
47
48 </beans>

```

3.3.3 스프링 설정파일 - boardContext.xml

WEB-INF/contexts/boardContext.xml : bean, p

```

01  <?xml version="1.0" encoding="UTF-8"?>
02  <beans xmlns="http://www.springframework.org/schema/beans"
03    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
04    xmlns:p="http://www.springframework.org/schema/p"
05    xsi:schemaLocation="http://www.springframework.org/schema/beans
06      http://www.springframework.org/schema/beans/spring-beans.xsd">
07
08  <!-- Dao -->
09  <bean id="boardDao" class="board.dao.BoardDaoImpl"
10    p:sqlSession-ref="sqlSessionTemplate" />
11
12  <!-- Model -->
13  <bean id="boardPaging" class="board.model.Paging" />
14
15  <!-- Controller -->
16  <!-- List 화면 -->
17  <bean id="BoardListController" class="board.controller.ListController"
18    p:boardListService-ref="boardListService" p:boardPaging-ref="boardPaging" />
19  <!-- 글쓰기 화면 -->
20  <bean id="BoardWriteController" class="board.controller.WriteController"
21    p:writeService-ref="writeService" />
22  <!-- 글보기 화면 -->
23  <bean id="BoardGetArticleController" class="board.controller.GetArticleController"
24    p:getArticleService-ref="getArticleService" />
25  <!-- 글 수정 -->

```

```

26 <bean id="BoardUpdateArticleController" class="board.controller.UpdateArticleController"
27   p:updateArticleService-ref="updateArticleService" />
28 <!-- 글 삭제 -->
29 <bean id="BoardDeleteArticleController" class="board.controller.DeleteArticleController"
30   p:deleteArticleService-ref="deleteArticleService" />
31
32 <!-- Service -->
33 <bean id="boardListService" class="board.service.BoardListServiceImpl"
34   p:boardDao-ref="boardDao" />
35 <bean id="writeService" class="board.service.WriteServiceImpl"
36   p:boardDao-ref="boardDao" />
37 <bean id="getArticleService" class="board.service.GetArticleServiceImpl"
38   p:boardDao-ref="boardDao" />
39 <bean id="updateArticleService" class="board.service.UpdateArticleServiceImpl"
40   p:boardDao-ref="boardDao" />
41 <bean id="deleteArticleService" class="board.service.DeleteArticleServiceImpl"
42   p:boardDao-ref="boardDao" />
43 </beans>

```

3.3.4 스프링 설정파일 - commentContext.xml

WEB-INF/contexts/commentContext.xml : bean, p	
01	<?xml version="1.0" encoding="UTF-8"?>
02	<beans xmlns="http://www.springframework.org/schema/beans"
03	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
04	xmlns:p="http://www.springframework.org/schema/p"
05	xsi:schemaLocation="http://www.springframework.org/schema/beans
06	http://www.springframework.org/schema/beans/spring-beans.xsd">
07	
08	<!-- Dao -->
09	<bean id="mybatisCommentDao" class="comment.dao.MybatisCommentDao"
10	p:sqlSession-ref="sqlSessionTemplate" />
11	
12	<!-- Controller -->
13	<bean id="CommentListController" class="comment.controller.CommentListController"

```

14    p:commentListImpl-ref="CommentListImpl" />
15    <bean id="CommentWriteController" class="comment.controller.CommentWriteController"
16      p:commentWriteImpl-ref="CommentWriteImpl" />
17    <bean id="CommentUpdateController" class="comment.controller.CommentUpdateController"
18      p:commentUpdateImpl-ref="CommentUpdateImpl" />
19    <bean id="CommentDeleteController" class="comment.controller.CommentDeleteController"
20      p:commentDeleteImpl-ref="CommentDeleteImpl" />
21
22    <!-- Service -->
23    <bean id="CommentListImpl" class="comment.service.CommentListImpl"
24      p:commentDao-ref="mybatisCommentDao" />
25    <bean id="CommentWriteImpl" class="comment.service.CommentWriteImpl"
26      p:commentDao-ref="mybatisCommentDao" />
27    <bean id="CommentUpdateImpl" class="comment.service.CommentUpdateImpl"
28      p:commentDao-ref="mybatisCommentDao" />
29    <bean id="CommentDeleteImpl" class="comment.service.CommentDeleteImpl"
30      p:commentDao-ref="mybatisCommentDao" />
31
32  </beans>

```

3.3.5 MyBatis 환경 설정 - mybatis-config.xml

src/mybatis-config.xml

```

01  <?xml version="1.0" encoding="UTF-8"?>
02  <!DOCTYPE configuration PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
03  "http://mybatis.org/dtd/mybatis-3-config.dtd">
04  <configuration>
05    <mappers>
06      <mapper resource="comment/dao/comment.xml" />
07      <mapper resource="board/dao/board.xml" />
08    </mappers>
09  </configuration>

```

3.3.6 스프링 서블릿 설정 - spring-servlet.xml

WEB-INF/spring-servlet.xml : bean, p, context

```
01 <?xml version="1.0" encoding="UTF-8"?>
02 <beans xmlns="http://www.springframework.org/schema/beans"
03   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
04   xmlns:context="http://www.springframework.org/schema/context"
05   xmlns:p="http://www.springframework.org/schema/p"
06   xsi:schemaLocation="http://www.springframework.org/schema/beans
07   http://www.springframework.org/schema/beans/spring-beans.xsd
08   http://www.springframework.org/schema/context
09   http://www.springframework.org/schema/context/spring-context-3.2.xsd">
10
11 <!-- HandlerMapping -->
12 <bean id="handlerMapping"
13   class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
14   <property name="mappings">
15     <value>
16       <!-- 게시판 -->
17       board/list.do=BoardListController
18       board/writeForm.do=BoardWriteController
19       board/content.do=BoardGetArticleController
20       board/updateForm.do=BoardUpdateArticleController
21       board/deleteForm.do=BoardDeleteArticleController
22
23       <!-- comment -->
24       comment/list.do=CommentListController
25       comment/write.do=CommentWriteController
26       comment/update.do=CommentUpdateController
27       comment/delete.do=CommentDeleteController
28     </value>
29   </property>
30 </bean>
31
32 </beans>
```

3.3.7 쿼리 mapper 설정 - board.xml

src/board/dao/board.xml

```
01 <?xml version="1.0" encoding="UTF-8"?>
02
03 <!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
04 "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
05 <mapper namespace="board.dao.board">
06   <resultMap id="BoardResult" type="board.model.BoardVO">
07     <result property="num" column="NUM" jdbcType="BIGINT" />
08     <result property="writer" column="WRITER" jdbcType="VARCHAR" />
09     <result property="emair" column="EMAIL" jdbcType="VARCHAR" />
10     <result property="subject" column="SUBJECT" jdbcType="VARCHAR" />
11     <result property="pass" column="PASS" jdbcType="VARCHAR" />
12     <result property="readcount" column="READCOUNT" jdbcType="BIGINT" />
13     <result property="regdate" column="REGDATE" jdbcType="TIMESTAMP" />
14     <result property="content" column="CONTENT" jdbcType="VARCHAR" />
15   </resultMap>
16
17   <!-- 전체 글 수 : 검 색 (A ||) -->
18   <select id="getAHCount" parameterType="java.util.Map"
19     resultType="Integer">
20     select count(*) from BOARD
21     where writer = #{search_text} or
22       subject like '%' || #{search_text} || '%' or
23       content like '%' || #{search_text} || '%'
24   </select>
25
26   <!-- 전체 글 수 : 검 색 (writer) -->
27   <select id="getWriterCount" parameterType="java.util.Map"
28     resultType="Integer">
29     select count(*) from BOARD where writer = #{search_text}
30   </select>
31
32   <!-- 전체 글 수 : 검색 (subject) -->
```

```

33 <select id="getSubjectCount" parameterType="java.util.Map"
34   resultType="Integer">
35   select count(*) from BOARD_DC
36   where subject like '%' || #{search_text} || '%'
37 </select>
38
39 <!-- 전체 글 수 : 검색(content) -->
40 <select id="getContentCount" parameterType="java.util.Map"
41   resultType="Integer">
42   select count(*) from BOARD_DC
43   where content like '%' || #{search_text} || '%'
44 </select>
45
46 <!-- List(목록보기) : 전체 -->
47 <select id="getList" parameterType="java.util.Map" resultMap="BoardResult">
48 <![CDATA[select * from (select rownum mum, num, writer, pass,
49 email, subject, content, regdate, readcount
50 from (select * from BOARD_DC order by num desc))
51 where mum >= #{starRow} and mum <= #{endRow}
52 ]]>
53 </select>
54
55 <!-- List(목록보기) : 검색(All) -->
56 <select id="getAllList" parameterType="java.util.Map" resultMap="BoardResult">
57 <![CDATA[select * from (select rownum mum, num, writer, pass, email,
58 subject, content, regdate, readcount
59 from (select * from BOARD_DC
60 where writer=#{search_text} or
61 subject like '%' || #{search_text} || '%' or
62 content like '%' || #{search_text} || '%'
63 order by num desc)
64 where mum >= #{starRow} and mum <= #{endRow}
65 ]]>
66 </select>

```

```
67
68    <!-- List(목록보기) : 검색(writer) -->
69    <select id="getWriterList" parameterType="java.util.Map"
70        resultMap="BoardResult">
71        <![CDATA[ select * from (select rownum rnum, num, writer, pass, email,
72            subject, content, regdate, readcount
73            from (select * from BOARD
74                where writer=#{search_text} order by num desc)
75            where rnum >= #{starRow} and rnum <= #{endRow}
76        ]]>
77        </select>
78
79    <!-- List(목록보기) : 검색(subject) -->
80    <select id="getSubjectList" parameterType="java.util.Map"
81        resultMap="BoardResult">
82        <![CDATA[
83            select * from (select rownum rnum, num, writer, pass, email,
84                subject, content, regdate, readcount
85                from (select * from BOARD
86                    where subject like '%' || #{search_text} || '%'
87                    order by num desc)
88            where rnum >= #{starRow} and rnum <= #{endRow}
89        ]]>
90        </select>
91
92    <!-- List(목록보기) : 검색(content) -->
93    <select id="getContentList" parameterType="java.util.Map"
94        resultMap="BoardResult">
95        <![CDATA[
96            select * from (select rownum rnum, num, writer, pass, email,
97                subject, content, regdate, readcount
98                from (select * from BOARD
99                    where content like '%' || #{search_text} || '%'
100                   order by num desc))
```

```

101 where num >= #{starRow} and num <= #{endRow}
102 ]]>
103     </select>
104
105     <!-- 글쓰기 -->
106     <insert id="insertArticle" parameterType="board.model.BoardVO">
107     <![CDATA[
108         insert into BOARD(num, writer, pass, email, subject,
109         content, regdate)
110         values (BOARD_seq.nextval,#{writer}, #{pass},
111             #{email},#{subject},#{content},#{regdate})
112     ]]>
113     </insert>
114     <!-- 글 보기 -->
115     <update id="upReadcount" parameterType="java.lang.Integer">
116     <![CDATA[
117         update BOARD set readcount = readcount+1
118         where num = #{num}
119     ]]>
120     </update>
121
122     <select id="getArticle" parameterType="java.lang.Integer"
123         resultMap="BoardResult">
124     <![CDATA[
125         select * from BOARD where num = #{num}
126     ]]>
127     </select>
128
129     <!-- 비밀번호 가져오기 -->
130     <select id="getPass" parameterType="java.lang.Integer"
131         resultType="String">
132     <![CDATA[
133         select pass from BOARD where num = #{num}
134     ]]>

```

```

135    </select>
136
137    <!-- 글 수정 -->
138    <update id="updateArticle" parameterType="board.model.BoardVO">
139    <![CDATA[
140        update BOARD_DC set writer = #{writer}, email=#{email},
141        subject=#{subject}, content=#{content}
142        where num=#{num}
143    ]]>
144    </update>
145
146    <!-- 글 삭제 -->
147    <delete id="deleteArticle" parameterType="java.lang.Integer">
148    <![CDATA[
149        delete from BOARD_DC where num=#{num}
150    ]]>
151    </delete>
152 </mapper>

```

3.3.8 쿼리 mapper 설정 - comment.xml

src/board/dao/comment.xml	
01	<?xml version="1.0" encoding="UTF-8"?>
02	
03	<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
04	"http://mybatis.org/dtd/mybatis-3-mapper.dtd">
05	<mapper namespace="comment.dao.comment">
06	<resultMap id="commentResult" type="comment.model.CommentVO">
07	<result property="commentId" column="COMMENT_ID" jdbcType="BIGINT" />
08	<result property="num" column="NUM" jdbcType="BIGINT" />
09	<result property="comment_name" column="COMMENT_NAME" jdbcType="VARCHAR" />
10	<result property="comment" column="COMMENTS" jdbcType="VARCHAR" />
11	<result property="regdate" column="REGDATE" jdbcType="TIMESTAMP" />
12	</resultMap>
13	

```

14 <!-- 전체글수 가져오기 -->
15 <select id="getListCount" resultType="Integer">
16   select count(*) from BOARD_DC_COMMENT
17 </select>
18
19 <select id="nextVal" resultType="Integer">
20   select BOARD_DC_COMMENT_SEQ.nextval from dual
21 </select>
22
23 <!-- 글가져오기 -->
24 <select id="getList" parameterType="java.util.Map" resultMap="commentResult">
25   select * from (select ROWNUM rnum, COMMENT_ID , NUM,
26   COMMENT_NAME, COMMENTS, REGDATE from
27   (select * from BOARD_DC_COMMENT
28   where NUM=#{num} order by COMMENT_ID ASC)
29   where rnum BETWEEN #{start} AND #{end}
30 </select>
31
32 <!-- insertMessage : 댓글쓰기 -->
33 <insert id="insertMessage" parameterType="comment.model.CommentVO">
34   insert into BOARD_DC_COMMENT (COMMENT_ID,NUM,
35   COMMENT_NAME, COMMENTS, REGDATE)
36   values(#{comment_id}, #{num}, #{comment_name},
37   #{comment},#{regdate})
38 </insert>
39
40 <!-- updateMessage : 댓글 수정 -->
41 <update id="updateMessage" parameterType="comment.model.CommentVO">
42   update BOARD_DC_COMMENT set COMMENTS = #{comment}
43   where COMMENT_ID = #{comment_id}
44 </update>
45
46 <!-- deleteMessage : 댓글삭제 -->
47 <delete id="deleteMessage" parameterType="Integer">

```

```
48    delete from BOARD_COMMENT  
49    where COMMENT_ID = #{comment_id}  
50    </delete>  
51  </mapper>
```

3.4 모델 클래스 작성

3.4.1 boardVO

src/board/model/BoardVO.java

```
01 package board.model;  
02  
03 import java.util.Date;  
04  
05 public class BoardVO {  
06     private int num;  
07     private String writer;  
08     private String email;  
09     private String subject;  
10     private String pass;  
11     private int readcount;  
12     private Date regdate;  
13     private String content;  
14     private String article_type;  
15     public int getNum() {  
16         return num;  
17     }  
18     public void setNum(int num) {  
19         this.num = num;  
20     }  
21     public String getWriter() {  
22         return writer;  
23     }  
24     public void setWriter(String writer) {  
25         this.writer = writer;
```

```
26 }
27 public String getEmail() {
28     return email;
29 }
30 public void setEmail(String email) {
31     this.email = email;
32 }
33 public String getSubject() {
34     return subject;
35 }
36 public void setSubject(String subject) {
37     this.subject = subject;
38 }
39 public String getPass() {
40     return pass;
41 }
42 public void setPass(String pass) {
43     this.pass = pass;
44 }
45 public int getReadcount() {
46     return readcount;
47 }
48 public void setReadcount(int readcount) {
49     this.readcount = readcount;
50 }
51 public Date getRegdate() {
52     return regdate;
53 }
54 public void setRegdate(Date regdate) {
55     this.regdate = regdate;
56 }
57 public String getContent() {
58     return content;
59 }
```

```

60 public void setContent(String content) {
61     this.content = content;
62 }
63 public String getArticle_type() {
64     return article_type;
65 }
66 public void setArticle_type(String article_type) {
67     this.article_type = article_type;
68 }
69 }
```

3.4.2 Paging.java

src/board/model/paging.java

```

01 package board.model;
02
03 public class Paging {
04     private Integer w_size; // 페이지당 보여줄 글수
05     private Integer p_size; // 페이지번호 표시수
06     private Integer writing_Count; // 전체 글수
07     private Integer cur_Page; // 현재 페이지
08     public Paging( ) {
09         super( );
10     }
11
12     /**
13      * Paging(한 화면에 보여질 글 수, 페 이 지 분할 수,
14      * 총 글의 갯수, 현재 보고 있는 페이지 번호 );
15      */
16     public void setPaging(Integer w_size, Integer p_size, Integer writing_Count, Integer cur_Page) {
17         this.w_size = w_size;
18         this.p_size = p_size;
19         this.writing_Count = writing_Count;
20         this.cur_Page = cur_Page;
21     }
```

```

22
23    /*
24     * 총 페이지 수를 리턴해주는 메소드 전체 글수를 페이지당 보여줄
25     * 글수로 나눈 값에 전체 글수를 페이지당 보여줄 글수로
26     * 나머지연산을 하여 구한 값이 0보다 크면 1을 더하고
27     * 아니면 0을 더함
28     * @return 총페이지수
29     */
30    public Integer getPage_Count() {
31        return (writing_Count / w_size) + (writing_Count % w_size > 0 ? 1 : 0);
32    }
33
34    /**
35     * 페이지 시작 수를 리턴해주는 메소드
36     * @return 페이지 시작수
37     */
38    public Integer getPage_Start() {
39        return ((cur_Page - 1) / p_size) * p_size + 1;
40    }
41
42    /*
43     * 페이지 마지막 수를 리턴해주는 메소드
44     * @return 페이지 마지막 수
45     */
46    public Integer getPage_End() {
47        return Math.min(getPage_Start() + p_size - 1, getPage_Count());
48    }
49
50    /*
51     * Pre 표시 여부
52     * @return boolean
53     */
54    public boolean isPre() {
55        return getPage_Start() != 1;

```

```
56    }
57
58    /*
59     * Next 표시 여부
60     * @return boolean
61     */
62    public boolean isNext( ) {
63        return getPage_End() < getPage_Count();
64    }
65
66    /*
67     * 글 범위 시작 번호
68     * @return 글 범위 시작 번호
69     */
70    public Integer getWriting_Start( ) {
71        return getWriting_End() - w_size;
72    }
73
74    /*
75     * 글 범위 끝 번호
76     * @return 글 범위 끝 번호
77     */
78    public Integer getWriting_End( ) {
79        return cur_Page * w_size;
80    }
81
82    public Integer getW_size( ) {
83        return w_size;
84    }
85
86    public void setW_size(Integer w_size) {
87        this.w_size = w_size;
88    }
89
```

```

90 public Integer getP_size() {
91     return p_size;
92 }
93
94 public void setP_size(Integer p_size) {
95     this.p_size = p_size;
96 }
97
98 public Integer getWriting_Count() {
99     return writing_Count;
100 }
101
102 public void setWriting_Count(Integer writing_Count) {
103     this.writing_Count = writing_Count;
104 }
105
106 public Integer getCur_Page() {
107     return cur_Page;
108 }
109
110 public void setCur_Page(Integer cur_Page) {
111     this.cur_Page = cur_Page;
112 }
113
114 }
```

3.5 게시판용 DAO

3.5.1 BoardDao.java

src/board/dao/boardDao.java	
01	package board.dao;
02	
03	import java.util.List;
04	import board.model.BoardVO;

```

05
06 public interface BoardDao {
07     int getListCount(Object obj); //전체 글 수 or 검색한 글 수
08     List<BoardVO> getList(Object obj); //전체 글 or 검색한 글 리스트
09     void insertArticle (BoardVO boarVo); //글 입력
10     BoardVO getArticle (Integer num); //글 가져오기
11     void updateArticle(BoardVO boardVo); //글 입력
12     String getPass(Integer num); //비밀번호 가져오기
13     void deleteArticle (Integer num); //글삭제
14 }

```

3.5.2 BoardDaoImpl.java

src/board/dao/boardDao.java

```

01 package board.dao;
02
03 import java.util.HashMap;
04 import java.util.List;
05 import java.util.Map;
06
07 import org.mybatis.spring.SqlSessionTemplate;
08
09 import board.model.BoardVO;
10
11 public class BoardDaoImpl implements BoardDao {
12     private SqlSessionTemplate sqlSession;
13
14     public void setSqlSession(SqlSessionTemplate sqlSession) {
15         this.sqlSession = sqlSession;
16     }
17
18     @Override
19     public List<BoardVO>getList(Object obj) {
20         Map<String, Object>map = new HashMap<String, Object>();
21         map = (Map<String, Object>) obj;

```

```
22
23     if (map.get("search_type").equals("all")) {
24         return sqlSession.selectList("board.dao.board.getAllList", obj);
25     } else if (map.get("search_type").equals("writer")) {
26         return sqlSession.selectList("board.dao.board.getWriterList", obj);
27     } else if (map.get("search_type").equals("subject")) {
28         return sqlSession.selectList("board.dao.board.getSubjectList", obj) ;
29     } else if (map.get("search_type").equals("content")) {
30         return sqlSession.selectList("board.dao.board.getContentList", obj);
31     } else {
32         return sqlSession.selectList("board.dao.board.getList", obj);
33     }
34 }
35
36 @Override
37 public int getListCount(Object obj) {
38     Map<String, Object> map = new HashMap<String, Object>();
39     map = (Map<String, Object>) obj;
40
41     if (map.get("search_type").equals("writer")) {
42         return sqlSession.selectOne("board.dao.board.getWriterCount", obj);
43     } else if (map.get("search_type").equals("subject")) {
44         return sqlSession.selectOne("board.dao.board.getSubjectCount", obj);
45     } else if (map.get("search_type").equals("content")) {
46         return sqlSession.selectOne("board.dao.board.getContentCount", obj);
47     } else {
48         return sqlSession.selectOne("board.dao.board.getAllCount", obj);
49     }
50 }
51
52 @Override
53 // 글쓰기
54 public void insertArticle(BoardVO boardVo) {
55     sqlSession.insert ("board.dao.board.insert Article", boardVo);
```

```

56 }
57
58 @Override
59 // 글보기
60 public BoardVO getArticle(Integer num) {
61     sqlSession.update("board.dao.board.upReadcount", num);
62     return sqlSession.selectOne("board.dao.board.getArticle", num);
63 }
64
65 // 글수정
66 @Override
67 public void updateArticle(BoardVO boardVo) {
68     sqlSession.update("board.dao.board.updateArticle", boardVo);
69 }
70
71 // 비밀번호 가져오기
72 @Override
73 public String getPass(Integer num) {
74     return sqlSession.selectOne("board.dao.board.getPass", num);
75 }
76
77 // 글 삭제
78 @Override
79 public void deleteArticle(Integer num) {
80     sqlSession.delete( "board.dao.board.delete Article", num);
81 }
82 }

```

3.6 게시판용 서비스

3.6.1 BoardListService.java

src/board/service/BoardListService.java	
01	package board.service;
02	

```
03 import java.util.List;
04
05 import board.model.BoardVO;
06
07 public interface BoardListService {
08     Integer getListCount(Object obj);
09     List<BoardVO> getBoardList(Object obj);
10 }
```

3.6.2 BoardListServiceImpl.java

src/board/service/BoardListServiceImpl.java

```
01 package board.service;
02
03 import java.util.List;
04
05 import board.dao.BoardDao;
06 import board.model.BoardVO;
07
08 public class BoardListServiceImpl implements BoardListService {
09     private BoardDao boardDao;
10
11     public void setBoardDao(BoardDao boardDao) {
12         this.boardDao = boardDao;
13     }
14
15     @Override
16     public List<BoardVO> getBoardList(Object obj) {
17         return this.boardDao.getList(obj);
18     }
19
20     @Override
21     public Integer getListCount(Object obj) {
22         return this.boardDao.getListCount(obj);
23     }
```

3.6.3 WriteService.java

src/board/service/WriteService.java

```
01 package board.service;  
02  
03 import board.model.BoardVO;  
04  
05 public interface WriteService {  
06     void insertWriting(BoardVO boardVo);  
07 }
```

3.6.4 WriteServiceImpl.java

src/board/service/WriteServiceImpl.java

```
01 package board.service;  
02  
03 import board.dao.BoardDao;  
04 import board.model.BoardVO;  
05  
06 public class WriteServiceImpl implements WriteService {  
07     private BoardDao boardDao;  
08  
09     public void setBoardDao(BoardDao boardDao) {  
10         this.boardDao = boardDao;  
11     }  
12  
13     @Override  
14     public void insertWriting(BoardVO boardVo) {  
15         this.boardDao.insertArticle(boardVo);  
16     }  
17 }
```

3.6.5 GetArticleService.java

```
src/board/service/GetArticleService.java
```

```
01 package board.service;  
02  
03 import board.model.BoardVO;  
04  
05 public interface GetArticleService {  
06     BoardVO getArticle(Integer num);  
07 }
```

3.6.6 GetArticleServiceImpl.java

```
src/board/service/GetArticleServiceImpl.java
```

```
01 package board.service;  
02  
03 import board.dao.BoardDao;  
04 import board.model.BoardVO;  
05  
06 public class GetArticleServiceImpl implements GetArticleService {  
07     private BoardDao boardDao;  
08     public BoardDao getBoardDao() {  
09         return boardDao;  
10     }  
11  
12     public void setBoardDao(BoardDao boardDao) {  
13         this.boardDao = boardDao;  
14     }  
15  
16     @Override  
17     public BoardVO getArticle(Integer num) {  
18         return this.boardDao.getArticle(num);  
19     }  
20 }
```

3.6.7 UpdateArticleService.java

src/board/service/UpdateArticleService.java

```
01 package board.service;  
02  
03 import board.model.BoardVO;  
04  
05 public interface UpdateArticleService {  
06     BoardVO getArticle(Integer num);  
07     String getPass(Integer num);  
08     void updateArticle(BoardVO boardVo);  
09 }
```

3.6.8 UpdateArticleServiceImpl.java

src/board/service/UpdateArticleServiceImpl.java

```
01 package board.service;  
02  
03 import board.dao.BoardDao;  
04 import board.model.BoardVO;  
05  
06 public class UpdateArticleServiceImpl implements UpdateArticleService {  
07     private BoardDao boardDao;  
08  
09     public void setBoardDao(BoardDao boardDao) {  
10         this.boardDao = boardDao;  
11     }  
12  
13     @Override  
14     public void updateArticle(BoardVO boardVo) {  
15         this.boardDao.updateArticle(boardVo);  
16     }  
17  
18     @Override  
19     public BoardVO getArticle(Integer num) {  
20         return this.boardDao.getArticle(num);  
21     }
```

```
22  
23     @Override  
24     public String getPass(Integer num) {  
25         return this.boardDao.getPass(num);  
26     }  
27 }
```

3.6.9 DeleteArticleService.java

src/board/service/DeleteArticleService.java

```
01 package board.service;  
02  
03 public interface DeleteArticleService {  
04     public String getPass(Integer num);  
05  
06     public void deleteArticle(Integer num);  
07 }
```

3.6.10 DeleteArticleServiceImpl.java

src/board/service/DeleteArticleServiceImpl.java

```
01 package board.service;  
02  
03 import board.dao.BoardDao;  
04  
05 public class DeleteArticleServiceImpl implements DeleteArticleService {  
06     private BoardDao boardDao;  
07  
08     public void setBoardDao(BoardDao boardDao) {  
09         this.boardDao = boardDao;  
10     }  
11     @Override  
12     public String getPass(Integer num) {  
13         return this.boardDao.getPass(num);  
14     }
```

```
15  
16     @Override  
17     public void deleteArticle(Integer num) {  
18         this.boardDao.deleteArticle(num);  
19     }  
20 }
```

3.7 게시판 컨트롤러 클래스

3.7.1 ListController.java

src/board/controller/ListController.java

```
01 package board.controller;  
02  
03 import java.util.HashMap;  
04 import java.util.List;  
05 import java.util.Map;  
06  
07 import javax.servlet.http.HttpServletRequest;  
08 import javax.servlet.http.HttpServletResponse;  
09  
10 import org.springframework.web.servlet.ModelAndView;  
11 import org.springframework.web.servlet.mvc.Controller;  
12  
13 import board.model.BoardVO;  
14 import board.model.Paging;  
15 import board.service.BoardListService;  
16  
17 public class ListController implements Controller {  
18     private BoardListService boardListService;  
19     private Paging boardPaging;  
20  
21     public void setBoardListService(BoardListService boardListService) {  
22         this.boardListService = boardListService;  
23     }
```

```
24  
25     public void setBoardPaging(Paging boardPaging) {  
26         this.boardPaging = boardPaging;  
27     }  
28  
29     @Override  
30     public ModelAndView handleRequest(  
31         HttpServletRequest request,  
32         HttpServletResponse response)  
33         throws Exception {  
34         request.setCharacterEncoding("utf-8");  
35  
36         //페이지 번호  
37         String pageNum = request.getParameter("pageNum");  
38         if (pageNum == null || pageNum == "") {  
39             pageNum = "1";  
40         }  
41  
42         //현재 페이지  
43         int currentPage = Integer.parseInt(pageNum);  
44         int pageSize = 10; //페이지당 보여줄 데이터수  
45         int pagenavi = 5; //페이지번호 보여줄수  
46  
47         //검색기능  
48         String search_type =  
49             request.getParameter("search_type");  
50         String search_text =  
51             request.getParameter("search_text");  
52         if(search_text != null) search_text =  
53             new String(search_text.getBytes("8859_1"), "utf—8");  
54         System.out.println(search_text);  
55         if(search_type == null) {search_type = ""};  
56         if(search_text == null) {search_text= ""};  
57         Map<String, Object> map =
```

```
58     new HashMap<String, Object>();
59     map.put("search_type", search_type);
60     map.put("search_text", search_text);
61
62     //List 처리
63     //전체 글 수 or 검색하고자 하는 글 수
64     int count = this.boardListService.getListCount(map);
65
66     //List에 보여지는 글번호
67     int number=count - (currentPage-1)*pageSize;
68     boardPaging.setPaging("// 페이징 클래스
69         pageSize, paginavi, count, currentPage);
70     map.put("starRow", boardPaging.getWriting_Start());
71     map.put("endRow", boardPaging.getWriting_End());
72
73     List<BoardVO> boardList =//전체글 or 검색하고자 하는 글
74         this.boardListService.getBoardList(map);
75     Map<String, Object> model =
76         new HashMap<String, Object>();
77     model.put("boardList",boardList);
78     model.put("count",count);
79     model.put("number",number);
80
81     model.put("search_text",search_text);
82     model.put("search_type",search_type);
83     model.put("pageNum",pageNum);
84     model.put("bp",boardPaging);
85
86     //해당 뷰에서 사용할 속성
87     ModelAndView modelAndView =
88         new ModelAndView();
89     modelAndView.setViewName("/board/list.jsp");
90     modelAndView.addAllObjects(model);
91     return modelAndView;
```

```
92    }
93 }
```

3.7.2 WriteController.java

```
src/board/controller/WriteController.java
```

```
01 package board.controller;
02
03 import java.text.SimpleDateFormat;
04 import java.util.Date;
05
06 import javax.servlet.http.HttpServletRequest;
07
08 import org.springframework.stereotype.Controller;
09 import org.springframework.validation.BindingResult;
10 import org.springframework.web.bind.annotation.RequestMapping;
11 import org.springframework.web.bind.annotation.RequestMethod;
12 import org.springframework.web.servlet.ModelAndView;
13
14 import board.model.BoardVO;
15 import board.service.WriteService;
16
17 @Controller
18 public class WriteController {
19     private WriteService writeService;
20
21     public void setWriteService(WriteService writeService) {
22         this.writeService = writeService;
23     }
24
25     @RequestMapping
26     public String setView( ) {
27         return "/board/writeForm.jsp";
28     }
29 }
```

```

30 @RequestMapping(method = RequestMethod.POST)
31 public ModelAndView onSubmit(HttpServletRequest request,
32     BoardVO boardVo,
33     BindingResult bindingResult) throws Exception {
34
35     //글쓰기 DB에 저장
36     SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
37     String date = sdf.format(new Date());
38     boardVo.setRegdate(new Date());
39     this.writeService.insertWriting(boardVo);
40     return new ModelAndView("redirect:/board/list.do");
41 }
42 }
```

3.7.3 GetArticleController.java

src/board/controller/GetArticleController.java

```

01 package board.controller;
02
03 import java.util.HashMap;
04 import java.util.Map;
05
06 import org.springframework.web.bind.annotation.RequestMapping;
07 import org.springframework.web.servlet.ModelAndView;
08
09 import board.model.BoardVO;
10 import board.service.GetArticleService;
11
12 public class GetArticleController {
13     private GetArticleService getArticleService;
14
15     public void setGetArticleService(GetArticleService getArticleService) {
16         this.getArticleService = getArticleService;
17     }
18 }
```

```

19
20     @RequestMapping
21     public ModelAndView getArticle (Integer num){
22         BoardVO boardVo = this.getArticleService.getArticle(num);
23         Map<String, Object> model = new HashMap<String, Object>();
24         model.put("vo", boardVo);
25         ModelAndView mav = new ModelAndView();
26         mav.setViewName("/board/content.jsp");
27         mav.addAllObjects(model);
28         return mav;
29     }
30 }
```

3.7.4 UpdateArticleController.java

src/board/controller/UpdateArticleController.java

```

01 package board.controller;
02
03 import java.util.HashMap;
04 import java.util.Map;
05
06 import javax.servlet.http.HttpServletRequest;
07
08 import org.springframework.stereotype.Controller;
09 import org.springframework.validation.BindingResult;
10 import org.springframework.web.bind.annotation.RequestMapping;
11 import org.springframework.web.bind.annotation.RequestMethod;
12 import org.springframework.web.servlet.ModelAndView;
13
14 import board.model.BoardVO;
15 import board.service.UpdateArticleService;
16
17 @Controller
18 public class UpdateArticleController {
19     private UpdateArticleService updateArticleService;
```

```
20
21 public void setUpdateArticleService (
22     UpdateArticleService updateArticleService) {
23     this.updateArticleService = updateArticleService;
24 }
25
26 @RequestMapping
27 public ModelAndView setView(Integer num){
28     BoardVO boardVo =
29         this.updateArticleService.getArticle(num);
30     Map<String, Object> model = new HashMap<String, Object>();
31     model.put("vo", boardVo);
32     ModelAndView mav = new ModelAndView();
33     mav.setViewName("/board/updateForm.jsp");
34     mav.addAllObjects(model);
35     return mav;
36 }
37
38 @RequestMapping(method=RequestMethod.POST)
39 public ModelAndView onSubmit(HttpServletRequest request,
40     BoardVO boardVo,
41     BindingResult bindingResult)throws Exception {
42     String pass=this.updateArticleService.getPass(
43         boardVo.getNum());
44     if(boardVo.getPass().equals(pass)){
45         this.updateArticleService.updateArticle(boardVo);
46         return new ModelAndView(
47             "redirect:/board/list.do");
48     } else {
49         boardVo = this.updateArticleService.getArticle(boardVo.getNum());
50         Map<String, Object> model = new HashMap<String, Object>();
51         model.put("vo", boardVo);
52         String passerror = "pass";
53         model.put("value", passerror);
```

```
54      ModelAndView mav = new ModelAndView();
55      mav.setViewName("/board/updateForm.jsp");
56      mav.addAllObjects(model);
57      return mav;
58  }
59 }
60 }
61 }
```

3.7.5 DeleteArticleController.java

src/board/controller/DeleteArticleController.java

```
01 package board.controller;
02
03 import org.springframework.stereotype.Controller;
04 import org.springframework.web.bind.annotation.RequestMapping;
05 import org.springframework.web.bind.annotation.RequestMethod;
06 import org.springframework.web.servlet.ModelAndView;
07
08 import board.model.BoardVO;
09 import board.service.DeleteArticleService;
10
11 @Controller
12 public class DeleteArticleController {
13     private DeleteArticleService deleteArticleService;
14
15     public void setDeleteArticleService(DeleteArticleService deleteArticleService) {
16         this.deleteArticleService = deleteArticleService;
17     }
18
19     @RequestMapping
20     public ModelAndView setView(Integer num){
21         ModelAndView mav = new ModelAndView("/board/deleteForm.jsp");
22         mav.addObject("num",num);
23         return mav;
```

```

24 }
25
26 @RequestMapping(method=RequestMethod.POST)
27 public ModelAndView onSubmit(Integer num, BoardVO boardVo) {
28     String dbpass= this.deleteArticleService.getPass(num);
29     if(boardVo.getPass().equals(dbpass)) {
30         this.deleteArticleService.deleteArticle (
31             boardVo.getNum());
32         ModelAndView mav = new ModelAndView("board/deleteForm.jsp");
33         mav.addObject("num",num);
34         mav.addObject("value", "successDelete");
35         return mav;
36     } else {
37         ModelAndView mav = new ModelAndView("/board/deleteForm.jsp");
38         mav.addObject("num",num);
39         mav.addObject("value", "passerror");
40         return mav;
41     }
42 }
43 }

```

3.8 뷰 페이지

3.8.1 style.css

WebContent/board/style.css

```

01 @CHARSET "UTF-8";
02
03 BODY {
04     FONT-SIZE: 9pt;
05     COLOR: black;
06     LINE-HEIGHT: 160%;
07     FONT-FAMILY: 굴림, verdana, tahoma
08 }
09
10 TD {

```

```
11 FONT-SIZE: 9pt;
12 COLOR: black;
13 LINE-HEIGHT: 160%;
14 FONT-FAMILY: 굴림, verdana, tahoma
15 }
16
17 SELECT {
18   FONT-SIZE: 9pt;
19   COLOR: black;
20   LINE-HEIGHT: 160%;
21   FONT-FAMILY: 굴림, verdana, tahoma
22 }
23
24 DIV {
25   FONT-SIZE: 9pt;
26   COLOR: black;
27   LINE-HEIGHT: 160%;
28   FONT-FAMILY: 굴림, verdana, tahoma
29 }
30
31 FORM {
32   FONT-SIZE: 9pt;
33   COLOR: black;
34   LINE-HEIGHT: 160%;
35   FONT-FAMILY: 굴림, verdana, tahoma
36 }
37
38 TEXTAREA {
39   BORDER-RIGHT: 1px solid #999999;
40   BORDER-TOP: 1px solid #999999;
41   FONT-SIZE: 9pt;
42   BORDER-LEFT: 1px solid #999999;
43   COLOR: BLACK;
44   BORDER-BOTTOM: 1px solid #999999;
```

```

45 FONT-FAMILY: 굴림, verdana;
46 BACKGROUND-COLOR: white
47 }
48
49 INPUT {
50 BORDER-RIGHT: 1px solid #999999;
51 BORDER-TOP: 1px solid #999999;
52 FONT-SIZE: 9pt;
53 BORDER-LEFT: 1px solid #999999;
54 COLOR: BLACK;
55 BORDER-BOTTOM: 1px solid #999999;
56 FONT-FAMILY: 굴림, verdana;
57 HEIGHT: 19px;
58 BACKGROUND-COLOR: white
59 }
60
61 A:link {
62 text-decoration: none;
63 color: #696969
64 }
65
66 A:hover {
67 text-decoration: yes;
68 color: #66CCFF
69 }
70
71 A:visited {
72 text-decoration: none;
73 color: #330066
74 }

```

3.8.2 color.jspf

WebContent/board/view/color.jspf

01	<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
----	-----------------------------------------------------------------

```

02 <cset var="bodyback_c" value="#FFEF5" />
03 <cset var="back_c" value="#8fbc8f7"/>
04 <cset var="title_c" value="#5f9ea07"/>
05 <cset var="value_c" value="#FFD1B7"/>
06 <cset var="bar" value="#778899"/>

```

3.8.3 list.jsp

WebContent/board/list.jsp

```

01 <?xml version="1.0" encoding="UTF-8"?>
02 <%@ page language="java" contentType="text/html; charset=UTF-8"
03   pageEncoding="UTF-8"%>
04 <%@ include file="view/colorjspf"%>
05 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
06 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
07 "http://www.w3.org/TR/xhtml/DTD/xhtml-transitional.dtd">
08 <html xmlns="http://www.w3.org/1999/xhtml">
09 <head>
10 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
11 <title>Insert title here</title>
12 <link href=".style.css" rel="stylesheet" type="text/css" />
13 <script type="text/javascript">
14 function search(){
15 var form = document.listForm;
16 if(form.search_text.value== null || form.search_text.value == ""){
17 alert("검색어를 입력해 주세요");
18 return;
19 }
20
21 var url = 'list.do?pageNum=1&search_type=' +
22 form.search_type.value + '&search_text=' +
23 encodeURIComponent(form.search_text.value);
24 window.location=url;
25 }
26 function passCheck(value) {

```

```

27 if(value == "successDelete" || value.equals("successDelete")) {
28     alert("삭제되었습니다.");
29 }
30 }
31 window.onload = passCheck("${value}");
32 </script>
33 </head>
34 <body bgcolor="${bodyback_c}">
35     <center>
36         <h3>게시판</h3>
37         <table width="700">
38             <tr>
39                 <td align="right" colspan="5"><input type="button" value="전체목록"
40                     onclick="window.location='list.do'" /> <input type="button"
41                     value="글쓰기" onclick="window.location='writeForm.do'" /></td>
42             </tr>
43             <tr bgcolor="${value_c}">
44                 <td align="center" width="60">번호</td>
45                 <td align="center" width="380">제목</td>
46                 <td align="center" width="100">작성자</td>
47                 <td align="center" width="100">작성일</td>
48                 <td align="center" width="60">조회수</td>
49             </tr>
50             <!-- 글이 없을 경우 -->
51             <c:if test="${count == 0 }">
52                 <tr>
53                     <td colspan="6" align="center">게시판에 저장된 글이 없습니다.</td>
54                 </tr>
55             </c:if>
56             <c:if test="${count != 0 }">
57                 <c:forEach var="vo" items="${boardList } ">
58                     <tr>
59                         <td align="center">${number}<c:set var="number"
60                             value="${number -1 }" scope="page" />
```

```

61      </td>
62      <td align="left"><a
63          href="content.do?num=${vo.num}&pageNum=${bp.cur_Page}">
64              ${vo.subject}</a> <c:if test="${vo.readcount > 10 }">
65                  </img>
66              </c:if></td>
67      <td align="center"><a href="mailto:${vo.email }">
68          ${vo.writer } </a></td>
69      <td align="center">${vo.regdate }</td>
70      <td align="center">${vo.readcount }</td>
71      </tr>
72  </c:forEach>
73  </c:if>
74  </table>
75  <br />
76  <c:if test="${bp.isPre( ) }">
77      <a href="list.do?pageNum=${bp.getPage_Start()-bp.p_size }">[이전]</a>
78  </c:if>
79  <c:forEach var="counter" begin="${bp.getPage_Start() }"
80      end="${bp.getPage.End() }">
81      <c:if test="${search_text ne '' } ">
82          <a href="javascript:window.location='list.do?pageNum=
83              ${ counter }&search_type=
84              ${ search_type }&search_text=${search_text }'">
85              [${counter}]</a>
86      </c:if>
87      <c:if test="${search_text eq '' } ">
88          <a href="javascript:window.location='list.do?pageNum=${counter }'">
89              [${ counter} ] </a>
90      </c:if>
91  </c:forEach>
92  <c:if test="${bp.isNext() }">
93      <a href="list.do?pageNum=${bp.getPage_Start()+bp.p_size }">[다음]</a>
94  </c:if>

```

```

95 <p></p>
96 <form method="get" name="listForm" action="list.do">
97   <input type="hidden" name="pageNum" value="${pageNum}" /> <select
98     name="search_type">
99     <option value="all" ${search_type == "all" ? "selected" : ""}>전체</option>
100    <option value="subject"
101      ${search_type == "subject" ? "selected" : ""}>제목</option>
102    <option value="writer"
103      ${search_type == "writer" ? "selected" : ""}>작성자</option>
104    <option value="content"
105      ${search_type == "content" ? "selected" : ""}>내용</option>
106  </select> <input type="text" name="search_text" value="${search_text}" /> <input
107    type="button" value="검색" onclick="search0()" />
108 </form>
109 </center>
110 </body>
111 </html>

```

3.8.4 writeForm.jsp

WebContent/board/writeForm.jsp

```

01 <?xml version="1.0" encoding="UTF-8" ?>
02 <%@ page language="java" contentType="text/html; charset=UTF-8"
03   pageEncoding="UTF-8"%>
04 <%@ include file="view/color.jspf"%>
05 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
06 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
07 <html xmlns="http://www.w3.org/1999/xhtml">
08 <head>
09 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
10 <title>Insert title here</title>
11 <link href="/style.css" rel="stylesheet" type="text/css" />
12 <script type="text/javascript" src="/js/script.js">
13
14 </script>

```

```

15 </head>
16 <body bgcolor="${bodyback_c }">
17 <div align="center" class="body">
18   <form method="post" name="writeForm" action="writeForm.do"
19     enctype="multipart.form-data">
20   <center>
21     <h3>글 쓰 기</h3>
22     <table width="450" border="1">
23       <tr>
24         <td align="center" bgcolor="${value_c }">작성자</td>
25         <td align="left"><input type="text" name="writer"
26           value="${writer }" /></td>
27       </tr>
28       <tr>
29         <td align="center" bgcolor="${value_c } ">비밀번호</td>
30         <td align="left" width="145"><input type="password"
31           name="pass" value="${pass }" /></td>
32       </tr>
33       <tr>
34         <td align="center" bgcolor="${value_c } ">이메일</td>
35         <td align="left"><input type="text" size="60" name="email"
36           value="${email }" /></td>
37       </tr>
38       <tr>
39         <td align="center" bgcolor="${value_c } ">제목</td>
40         <td align="left"><input type="text" size="60" name="subject"
41           value="${subject }" /></td>
42       </tr>
43       <tr>
44         <td align="center" bgcolor="${value_c } ">내용</td>
45         <td align="left"><textarea name="content" rows="13" cols="60"> </textarea></td>
46       </tr>
47     </table>
48     <br /> <input type="button" value="글쓰기" onclick="writeCheck()" />

```

```

49      <input type="button" value="목록보기"
50          onclick="window.location='list.do'" />
51      </center>
52  </form>
53  </div>
54 </body>
55 </html>

```

3.8.5 script.js

WebContent/js/script.js

```

01 function writeCheck(){
02     if (document.writeForm.writer.value == ""){
03         alert("작성자를 입력해 주세요");
04         document.write.writer.focus();
05         return;
06     }
07
08     if (document.writeForm.pass.value == ""){
09         alert("비밀번호를 입력해 주세요");
10         document.write.pass.focus();
11         return;
12     }
13
14     if (document.writeForm.email.value == ""){
15         alert("이메일을 입력해 주세요");
16         document.write.email.focus();
17         return;
18     }
19
20     if (document.writeForm.subject.value == ""){
21         alert("제목을 입력해 주세요");
22         document.write.subject.focus();
23         return;
24     }

```

```

25 if (document.writeForm.content.value == ""){
26     alert("내용을 입력해 주세요");
27     document.write.subject.focus();
28     return;
29 }
30
31 document.writeForm.submit();
32 }
33
34
35 function addFile(num) {
36     var v=document.getElementById('list_dis');
37     if(num == "b"){
38         v.style.display = "";
39     }
40     if(num == "a"){
41         v.style.display = "none";
42     }
43 }

```

3.8.6 content.jsp

WebContent/board/content.jsp

```

01 <?xml version="1.0" encoding="UTF-8"?>
02 <%@ page language="java" contentType="text/html; charset=UTF-8"
03     pageEncoding="UTF-8"%>
04 <%@ include file="view/color.jspf"%>
05 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
06 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
07 <html xmlns="http://www.w3.org/1999/xhtml">
08 <head>
09 <title>Insert title here</title>
10 <link href=".style.css" rel="stylesheet" type="text/css" />
11 <script type="text/javascript" src=".comment/js/ajax.js"></script>
12 <script type="text/javascript" src=".comment/js/comment.js"></script>

```

```

13 <script type="text/javascript">
14 window.onload = function () {
15     loadCommentList(document.writeForm.num.value);
16 }
17
18 function makeCommentView(comment) {
19     var commentDiv = document.createElement('div');
20     commentDiv.setAttribute('id', 'c' + comment.id);
21     var html = '<table width="700" height="50"><tr><td><strong>' +
22     comment.name +
23     '</strong>&nbsp;&nbsp;&nbsp;' +
24     comment.content.replace(/\n/g, '\n<br/>') +
25     '&nbsp;&nbsp;&nbsp;' +
26     '<FONT color="#FF0000"' +
27     + comment.regdate + '</FONT>' +
28     '&nbsp;&nbsp;&nbsp;' +
29     '+ '<input type="button" value="수정" onclick= "viewUpdateForm(' +
30     + comment.id + ')"/>' + '&nbsp;&nbsp;' +
31     '+ '<input type="button" value="삭제" onclick= "confirmDeletion(' +
32     + comment.id + ')"/></td></tr></table>';
33     commentDiv.innerHTML = html;
34     commentDiv.comment = comment;
35     commentDiv.className = "comment";
36     return commentDiv;
37 }
38 </script>
39 </head>
40 <body bgcolor="${bodyback_c}">
41     <div align="center" class="body">
42         <form method="post" name="writeForm">
43             <input type="hidden" name="num" value="${vo.num }" />
44             <center>
45                 <h3>글보기</h3>
46                 <table width="600" border="1">

```

```

47      <tr>
48          <td align="center" width="100" bgcolor="#${value_c}">번호</td>
49          <td align="left" width="200">${vo.num}</td>
50          <td align="center" width="100" bgcolor="#${value_c}">작성일</td>
51          <td align="left" width="200">${vo.regdate}</td>
52      </tr>
53      <tr>
54          <td align="center" bgcolor="#${value_c}">작성자</td>
55          <td align="left">${vo.writer}</td>
56          <td align="center" bgcolor="#${value_c}">조회수</td>
57          <td align="left">${vo.readcount}</td>
58      <tr>
59          <td align="center" bgcolor="#${value_c}">이메일</td>
60          <td colspan="7" align="left">${vo.email}</td>
61      </tr>
62      <tr>
63          <td align="center" bgcolor="#${value_c}">제목</td>
64          <td colspan="7" align="left">${vo.subject}</td>
65      </tr>
66      <tr>
67          <td align="center" bgcolor="#${value_c}">내용</td>
68          <td colspan="7" align="left" height="100"><pre>${vo.content}</pre>
69          </td>
70      </tr>
71  </table>
72  <br /> <input type="button" value="글수정"
73  onclick="window.location='updateForm.do?num=${vo.num}'" /> <input
74  type="button" value="글삭제"
75  onclick="window.location='deleteForm.do?num=${vo.num}'" /> <input
76  type="button" value="목록보기" onclick="window.location='list.do'" />
77  </center>
78  </form>
79  </div>
80  <div id="commentList" align="center" class="body"></div>

```

```

81 <div id="commentAdd" align="center">
82     <!-- 댓글 입력 폼 -->
83     <form action="" name="addForm" method="post">
84         <input type="hidden" name="num" value="${vo.num }" /> 이 룸 : <input
85             type="text" name="name" size="10"> </input> 내 용 :
86             <textarea name="content" cols="20" rows="2"> </textarea>
87             <input type="button" value="등록" onclick="addComment()" />
88         </form>
89     </div>
90     <div id="commentUpdate" style="display: none">
91         <form action="" name="updateForm">
92             <input type="hidden" name="id" value="" /> <input type="hidden"
93                 name="regdate" value="" /> 이 룸 : <input type="text" name="name"
94                 size="1"></input> <br /> 내 용 :
95                 <textarea name="content" cols="20" rows="2">
96             </textarea>
97             <br /> <input type="button" value="등록" onclick="updateComment()" />
98             <input type="button" value="취소" onclick="cancelUpdate()" />
99         </form>
100    </div>
101    </body>
102 </html>

```

3.8.7 updateForm.jsp

WebContent/board/updateForm.jsp

```

01 <?xml version="1.0" encoding="UTF-8"?>
02 <%@ page language="java" contentType = "text/html; charset=UTF-8"
03 pageEncoding="UTF-8"%>
04 <%@ taglib prefix="c" uri="http://javasun.com/jsp/jstl/core"%>
05 <%@ include file="view/color.jspf" %>
06 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
07 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
08 <html xmlns="http://www.w3.org/1999/xhtml">
09 <head>

```

```

10 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
11 <title> Insert title here </title>
12 <link href=".style.css" rel="stylesheet" type="text/css" />
13 <script type="text/javascript" src=".js/script.js" > </script>
14 <script type="text/javascript">
15 function passCheck(value) {
16 if (value == "pass" || value.equals("pass")) {
17 alert("비밀번호가 틀렸습니다.");
18 }
19 }
20 window.onload = passCheck("${value}");
21 </script>
22 </head>
23 <body bgcolor ="${bodyback_c}">
24 <div align="center" class="body">
25 <form method="post" name="writeForm" action="updateForm.do?num=${vo.num }">
26 <center>
27 <h3>글 수 정 </h3>
28 <table width="450" border="1">
29 <tr>
30 <td align="center" width="80" bgcolor ="${value_c }">작성자</td>
31 <td align="left"> <input type="text" size="22" name="writer" value="${vo.writer }" /> </td>
32 </tr>
33 <tr>
34 <td width= = "70" bgcolor ="${value_c }" align="center">
35 비밀번호 </td>
36 <td align="left"><input type="password" name="pass" value="${pass }" /> </td>
37 </tr>
38 <tr>
39 <td align="center" bgcolor ="${value_c }">이메일</td>
40 <td align="left" > <input type="text" size = "61" name = "email" value = "${vo.email }" /> </td>
41 </tr>
42 <tr>
43 <td align= "center" bgcolor ="${value_c }">제목 </td>

```

```

44 <td align= "left"><input type="text" size="61" name="subject" value="${vo.subject }" /> </td>
45 </tr>
46 <tr>
47 <td align= "center" bgcolor ="${value_c }" >내용 </td>
48 <td> <textarea name = "content" rows="13" cols="61">
49 ${vo.content } </textarea></td>
50 </tr>
51 </table>
52 <br /> <input type = "button" value="글수정" onclick ="writeCheck()" />
53 <input type="button" value = "목록보기" onclick="window.location='list.do'" />
54 </center>
55 </form>
56 </div>
57 </body>
58 </html>

```

3.8.8 deleteForm.jsp

WebContent/board/deleteForm.jsp

```

01 <?xml version="1.0" encoding="UTF-8"?>
02 <%@ page language="java" contentType="text/html; charset=UTF-8"
03 pageEncoding="UTF-8"%>
04 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
05 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
06 <html xmlns="http://www.w3.org/1999/xhtml">
07 <head>
08 <meta http-equiv="Content-Type" content="text/html; charset=UTF— 8" />
09 <title>글삭제</title>
10 <script type="text/javascript">
11     function passCheck(value) {
12         if (value == "successDelete") {
13             alert("글이 삭제되었습니다.");
14
15             window.location = "list.do";
16     }

```

```

17     if (value == "passerror") {
18
19         alert("비밀번호가 틀렸습니다.");
20
21     }
22
23 }
24
25
26 window.onload = passCheck("${ value}");
27 </script>
28 </head>
29 <body>
30     <form method="post" action="deleteForm.do?num=${num }">
31         <table>
32             <tr>
33                 <td>글 을 삭제하시려면 비밀번호를 입력해주세요.</td>
34             </tr>
35             <tr>
36                 <td><input type="password" name="pass" value="${pass}" /></td>
37                 <td><input type="submit" value="확인" /></td>
38             </tr>
39         </table>
40     </form>
41 </body>
42 </html>

```

3.9 댓글 관련 모델 클래스

3.9.1 CommentVO.java

src/comment/model/CommentVO.java

```

01 package comment.model;
02
03 import java.util.Date;
04

```

```
05 public class CommentVO {  
06     private int commented;  
07     private int num;  
08     private String comment_name;  
09     private String comment;  
10     private Date regdate;  
11     public int getCommented() {  
12         return commented;  
13     }  
14     public void setCommented(int commented) {  
15         this.commented = commented;  
16     }  
17     public int getNum() {  
18         return num;  
19     }  
20     public void setNum(int num) {  
21         this.num = num;  
22     }  
23     public String getComment_name() {  
24         return comment_name;  
25     }  
26     public void setComment_name(String comment_name) {  
27         this.comment_name = comment_name;  
28     }  
29     public String getComment() {  
30         return comment;  
31     }  
32     public void setComment(String comment) {  
33         this.comment = comment;  
34     }  
35     public Date getRegdate() {  
36         return regdate;  
37     }  
38     public void setRegdate(Date regdate) {
```

```
39     this.regdate = regdate;
40 }
41 }
```

3.10 댓글 관련 DAO 클래스

3.10.1 CommentDao.java

src/comment/dao/CommentDao.java

```
01 package comment.dao;
02
03 import java.util.List;
04
05 import comment.model.CommentVO;
06
07 public interface CommentDao {
08     public int count();
09     public int nextVal();
10     public List<CommentVO> select(int begin, int end, int num);
11     public void insert(CommentVO message);
12     public void delete(int id);
13     public void update(CommentVO message);
14 }
```

3.10.2 MybatisCommentDao.java

src/comment/dao/MybatisCommentDao.java

```
01 package comment.dao;
02
03 import java.util.HashMap;
04 import java.util.List;
05 import java.util.Map;
06
07 import org.mybatis.spring.SqlSessionTemplate;
08
09 import comment.model.CommentVO;
```

```
10
11 public class MybatisCommentDao implements CommentDao {
12     private SqlSessionTemplate sqlSession;
13     public void setSqlSession(SqlSessionTemplate sqlSession) {
14         this.sqlSession = sqlSession;
15     }
16
17     @Override
18     public int count() {
19         return sqlSession.selectOne("comment.dao.comment.getListCount");
20     }
21
22     @Override
23     public List<CommentVO> select(int begin, int end, int num) {
24         Map<String, Object> param = new HashMap<String, Object>();
25         param.put("start", begin);
26         param.put("end", end);
27         param.put("num", num);
28         return sqlSession.selectList("comment.dao.comment.getList", param);
29     }
30
31     @Override
32     public void delete (int id) {
33         sqlSession.delete("comment.dao.comment.deleteMessage", id);
34     }
35
36     @Override
37     public int nextVal() {
38         return sqlSession.selectOne("comment.dao.comment.nextVal");
39     }
40
41     @Override
42     public void insert(CommentVO message) {
43         sqlSession.insert("comment.dao.comment.insertMessage", message);
```

```
44 }
45
46     @Override
47     public void update (CommentVO message) {
48         sqlSession.update("comment.dao.comment.updateMessage", message);
49     }
50 }
```

3.11 댓글 관련 서비스 클래스

3.11.1 CommentListService

src/comment/service/CommentListService.java

```
01 package comment.service;
02
03 import java.util.List;
04
05 import comment.model.CommentVO;
06
07 public interface CommentListService {
08     List<CommentVO> getMessageList(int pageNum, int num);
09 }
```

3.11.2 CommentListImpl.java

src/comment/service/CommentListImpl.java

```
01 package comment.service;
02
03 import java.util.List;
04
05 import org.springframework.transaction.annotation.Transactional;
06
07 import comment.dao.CommentDao;
08 import comment.model.CommentVO;
09
10 public class CommentListImpl implements CommentListService {
```

```
11 public static final int DEFAULT_PAGE_SIZE = 20;
12 private int pageSize = DEFAULT_PAGE_SIZE;
13 private CommentDao commentDao;
14
15 public void setCommentDao(CommentDao commentDao) {
16     this.commentDao = commentDao;
17 }
18
19 @Transactional
20 public List<CommentVO> getMessageList(int pageNum, int num) {
21     int totalCount = commentDao.count();
22     return commentDao.select(1, totalCount, num);
23 }
24 public int getPageSize() {
25     return pageSize;
26 }
27
28 public void setPageSize(int pageSize) {
29     this.pageSize = pageSize;
30 }
31 }
```

3.11.3 CommentWriteService

src/comment/service/CommentWriteService.java

```
01 package comment.service;
02
03 import comment.model.CommentVO;
04
05 public interface CommentWriteService {
06     int nextVal();
07     CommentVO write(CommentVO message);
08 }
```

3.11.4 CommentWriteImpl.java

src/comment/service/CommentWriteImpl.java

```
01 package comment.service;
02
03 import org.springframework.transaction.annotation.Transactional;
04
05 import comment.dao.CommentDao;
06 import comment.model.CommentVO;
07
08 public class CommentWriteImpl implements CommentWriteService {
09     private CommentDao commentDao;
10     public void setCommentDao(CommentDao commentDao) {
11         this.commentDao = commentDao;
12     }
13
14     @Transactional
15     public CommentVO write (CommentVO message) {
16         commentDao.insert(message);
17
18         return message;
19     }
20
21     @Override
22     public int nextVal() {
23         return commentDao.nextVal();
24     }
25 }
```

3.11.5 CommentUpdateService

src/comment/service/CommentUpdateService.java

```
01 package comment.service;
02
03 import comment.model.CommentVO;
04
05 public interface CommentUpdateService {
```

```
06 void update(CommentVO message);  
07 }
```

3.11.6 CommentUpdateImpl.java

src/comment/service/CommentUpdateImpl.java

```
01 package comment.service;  
02  
03 import comment.dao.CommentDao;  
04 import comment.model.CommentVO;  
05  
06 public class CommentUpdateImpl implements CommentUpdateService {  
07     private CommentDao commentDao;  
08     public void setCommentDao(CommentDao commentDao) {  
09         this.commentDao = commentDao;  
10     }  
11  
12     @Override  
13     public void update(CommentVO message) {  
14         commentDao.update (message);  
15     }  
16 }
```

3.11.7 CommentUpdateService

src/comment/service/CommentUpdateService.java

```
01 package comment.service;  
02  
03 public interface CommentDeleteService {  
04     void delete(int id);  
05 }
```

3.11.8 CommentUpdateImpl.java

src/comment/service/CommentUpdateImpl.java

```
01 package comment.service;
```

```
02  
03 import comment.dao.CommentDao;  
04  
05 public class CommentDeleteImpl implements CommentDeleteService {  
06     private CommentDao CommentDao;  
07  
08     public void setCommentDao(CommentDao CommentDao) {  
09         this.CommentDao = CommentDao;  
10     }  
11  
12     @Override  
13     public void delete(int id) {  
14         CommentDao.delete(id);  
15     }  
16 }
```

3.12 댓글 관련 컨트롤러

3.12.1 CommentListController

src/comment/controller/CommentListController.java

```
01 package comment.controller;  
02  
03 import java.sql.SQLException;  
04 import java.util.HashMap;  
05 import java.util.List;  
06 import java.util.Map;  
07  
08 import javax.servlet.http.HttpServletRequest;  
09 import javax.servlet.http.HttpServletResponse;  
10  
11 import org.springframework.web.servlet.ModelAndView;  
12 import org.springframework.web.servlet.mvc.Controller;  
13  
14 import comment.model.CommentVO;  
15 import comment.service.CommentListImpl;
```

```

16
17 public class CommentListController implements Controller {
18     private CommentListImpl commentListImpl;
19     public void setCommentListImpl(CommentListImpl commentListImpl) {
20         this.commentListImpl = commentListImpl;
21     }
22
23     @Override
24     public ModelAndView handleRequest(HttpServletRequest request, HttpServletResponse
25 response) throws SQLException {
26         String num = request.getParameter("num");
27         List<CommentVO> list = commentListImpl.getMessageList(1, Integer.parseInt(num));
28
29         //모델 생성
30         Map<String, Object> model = new HashMap<String, Object>();
31         model.put("list", list);
32
33         //반환값인 ModelAndView 인스턴스 생성
34         ModelAndView modelAndView = new ModelAndView();
35         modelAndView.setViewName("/comment/list.jsp");
36         modelAndView.addAllObjects(model);
37         return modelAndView;
38     }
39 }
```

3.12.2 CommentWriteController

src/comment/controller/CommentWriteController.java

```

01 package comment.controller;
02
03 import java.util.Date;
04 import java.util.HashMap;
05 import java.util.Map;
06
07 import javax.servlet.http.HttpServletRequest;
```

```
08 import javax.servlet.http.HttpServletResponse;
09
10 import org.springframework.web.servlet.ModelAndView;
11 import org.springframework.web.servlet.mvc.Controller;
12
13 import comment.model.CommentVO;
14 import comment.service.CommentWriteImpl;
15
16 public class CommentWriteController implements Controller {
17     private CommentWriteImpl CommentWriteImpl;
18     private CommentVO comment;
19
20     public void setCommentWriteImpl(CommentWriteImpl CommentWriteImpl) {
21         this.CommentWriteImpl = CommentWriteImpl;
22     }
23     public ModelAndView handleRequest(HttpServletRequest request, HttpServletResponse
24 response) throws Exception {
25         request.setCharacterEncoding("UTF-8");
26         comment = new CommentVO();
27         comment.setComment_id(CommentWriteImpl.nextVal());
28         comment.setNum(Integer.parseInt(request.getParameter("num")));
29         comment.setComment_name(request.getParameter("name"));
30         comment.setComment(request.getParameter("content"));
31         comment.setRegdate(new Date());
32         CommentWriteImpl.write(comment);
33
34         // 모델 생성
35         Map<String, Object> model = new HashMap<String, Object>();
36         model.put("comment", comment);
37
38         // 반환값인 ModelAndView 인스턴스 생성
39         ModelAndView modelAndView = new ModelAndView();
40         modelAndView.setViewName("/comment/write.jsp");
41         modelAndView.addAllObjects(model);
```

```
42     return modelAndView;
43 }
44 }
```

3.12.3 CommentUpdateController

src/comment/controller/CommentUpdateController.java

```
01 package comment.controller;
02
03 import java.text.DateFormat;
04 import java.util.Date;
05 import java.util.HashMap;
06 import java.util.Map;
07
08 import javax.servlet.http.HttpServletRequest;
09 import javax.servlet.http.HttpServletResponse;
10
11 import org.springframework.web.servlet.ModelAndView;
12 import org.springframework.web.servlet.mvc.Controller;
13
14 import comment.model.CommentVO;
15 import comment.service.CommentUpdateImpl;
16
17 public class CommentUpdateController implements Controller {
18     private CommentUpdateImpl commentUpdateImpl;
19     private CommentVO comment;
20     public void setCommentUpdateImpl(CommentUpdateImpl commentUpdateImpl) {
21         this.commentUpdateImpl = commentUpdateImpl;
22     }
23
24     @Override
25     public ModelAndView handleRequest(HttpServletRequest request, HttpServletResponse
26 response) throws Exception {
27         request.setCharacterEncoding("UTF-8");
28     }
}
```

```

29   DateFormat df = DateFormat.getDateInstance();
30   String date = request.getParameter("regdate");
31
32   comment = new CommentVO();
33   comment.setComment_id(Integer.parseInt(request.getParameter("id")));
34   comment.setComment_name(request.getParameter("name"));
35   comment.setComment(request.getParameter("content"));
36
37   String[] dates = date.split(" ");
38   String[] ymd = dates[0].split("-");
39   String[] hms = dates[1].split(":");
40
41   Date myDate = new Date();
42   myDate.setYear( Integer.parseInt(ymd[0]));
43   myDate.setMonth( Integer.parseInt(ymd[1]) - 1);
44   myDate.setDate( Integer.parseInt(ymd[2]));
45
46   myDate.setHours( Integer.parseInt(hms[0]));
47   myDate.setMinutes ( Integer.parseInt(hms[1]));
48   myDate.setSeconds( Integer.parseInt(hms[2]));
49
50   comment.setRegdate(myDate);
51   CommentUpdateImpl.update(comment);
52
53   // 모델 생성
54   Map<String, Object> model = new HashMap<String, Object>();
55   model.put("comment", comment);
56
57   // 반환값인 ModelAndView 인스턴스 생성
58   ModelAndView modelAndView = new ModelAndView();
59   modelAndView.setViewName("/comment/update.jsp");
60   modelAndView.addAllObjects(model);
61   return modelAndView;
62 }
```

3.12.4 CommentDeleteController

src/comment/controller/CommentDeleteController.java

```
01 package comment.controller;
02
03 import java.util.HashMap;
04 import java.util.Map;
05
06 import javax.servlet.http.HttpServletRequest;
07 import javax.servlet.http.HttpServletResponse;
08
09 import org.springframework.web.servlet.ModelAndView;
10 import org.springframework.web.servlet.mvc.Controller;
11
12 import comment.service.CommentDeleteImpl;
13
14 public class CommentDeleteController implements Controller {
15     private CommentDeleteImpl CommentDeleteImpl;
16     public void setCommentDeleteImpl(CommentDeleteImpl CommentDeleteImpl) {
17         this.CommentDeleteImpl = CommentDeleteImpl;
18     }
19
20     @Override
21     public ModelAndView handleRequest(HttpServletRequest request, HttpServletResponse
22 response) throws Exception {
23         int id = Integer.parseInt(request.getParameter("id"));
24         CommentDeleteImpl.delete(id);
25
26         // 모델 생성
27         Map<String, Object> model = new HashMap<String, Object>();
28         model.put("id", id);
29
30         // 반환값인 ModelAndView 인스턴스 생성
```

```

31     ModelAndView modelAndView = new ModelAndView();
32     modelAndView.setViewName("/comment/delete.jsp");
33     modelAndView.addAllObjects(model);
34     return modelAndView;
35   }
36 }
```

3.13 댓글 관련 화면

3.13.1 ajax.js

WebContent/comment/js/ajax.js

```

01 var ajax = {};
02 ajax.xhr = {};
03 ajax.xhr.Request == function(url, params, callback, method) {
04   this.url == url;
05   this.params = params;
06   this.callback = callback;
07   this.method = method;
08   this.send();
09 }
10
11 function log(msg) {
12   var console = document.getElementById("debugConsole");
13   if (console != null) {
14     console.innerHTML += msg + "<br/> ";
15   }
16 }
17
18 var req = null;
19 ajax.xhr.Request.prototype = {
20   getXMLHttpRequest : function() {
21     if (window.ActiveXObject) {
22       try {
23         return new ActiveXObject("Msxml2.XMLHTTP");
24       } catch (e) {
```

```

25     try {
26         return new ActiveXObject("Microsoft.XMLDOM");
27     } catch ( el ) {
28         return null;
29     }
30 }
31 } else if (window.XMLHttpRequest) {
32     return new XMLHttpRequest();
33 } else {
34     return null;
35 }
36 },
37 send : function() {
38     this.req = this.getXMLHttpRequest();
39     var httpMethod = this.method ? this.method : 'GET';
40     if (httpMethod != 'GET' && httpMethod != 'POST') {
41         httpMethod = 'GET';
42     }
43     var httpParams = (this.params == null || this.params == "") ? null : this.params;
44     var httpUrl = this.url;
45
46     if (httpMethod == 'GET' && httpParams != null) {
47         httpUrl = httpUrl + "?" + httpParams;
48     }
49
50     this.req.open(httpMethod, httpUrl, true);
51     this.req.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
52     var request = this;
53     this.req.onreadystatechange = function() {
54         request.onStateChange.call(request);
55     };
56     this.req.send(httpMethod == 'POST' ? httpParams : null);
57 },
58 onStateChange : function() {

```

```
59     this.callback(this.req) ;  
60 }  
61 }
```

3.13.2comment.js

WebContent/comment/js/comment.js

```
01 function loadCommentList(num) {  
02     var params = "num=" + encodeURIComponent(num);  
03     new ajax.xhr.Request("/board_comment/comment/list.do", params, loadCommentResult, 'GET');  
04 }  
05  
06 function loadCommentResult(req) {  
07     if (req.readyState == 4) {  
08         if (req.status == 200) {  
09             var xmlDoc = req.responseXML;  
10             var code = xmlDoc.getElementsByTagName('code').item(0).firstChild.nodeValue;  
11  
12             if (code == 'success') {  
13                 var commentList = eval("(" + xmlDoc.getElementsByTagName('data').item(0).firstChild.nodeValue + ")");  
14                 var listDiv = document.getElementById('commentList');  
15                 for (var i=0; i<commentList.length; i++) {  
16                     var commentDiv = makeCommentView (commentList[i]);  
17                     listDiv.appendChild(commentDiv);  
18                 }  
19             } else if (code == 'error') {  
20                 var message = xmlDoc.getElementsByTagName('message').item(0).firstChild.nodeValue;  
21                 alert("에러 발생:" + message);  
22             }  
23         } else {  
24             alert("댓글 목록 로딩 실패:" + req.status);  
25         }  
26     }  
27 }  
28 }
```

```

29 function addComment() {
30     var num = document.addForm.num.value;
31     var name = document.addForm.name.value;
32     var content = document.addForm.content.value;
33
34     if(!name){
35         alert("이름을 입력해주세요!");
36         document.addForm.name.focus();
37         return;
38     }
39     if(!content) {
40         alert("내용을 입력해주세요!");
41         document.addForm.content.focus();
42         return;
43     }
44
45     var params = "num=" + encodeURIComponent(num) + "&" +
46     "name = " + encodeURIComponent(name) + "&" + "content=" +
47     + encodeURIComponent (content);
48
49     new ajax.xhr.Request('/board.comment/comment/write.do', params, addResult, 'POST');
50 }
51
52 function addResult(req) {
53     if (req.readyState == 4) {
54         if (req.status == 200) {
55             var xmlDoc = req.responseXML;
56             var code = xmlDoc.getElementsByTagName('code').item(0).firstChild.nodeValue;
57
58             if (code == 'success') {
59                 var comment = eval("(" +
60                 xmlDoc.getElementsByTagName('data').item(0).firstChild.nodeValue + ")");
61
62                 var listDiv = document.getElementById('commentList');

```

```

63     var commentDiv = makeCommentView(comment);
64     listDiv.appendChild(commentDiv);
65
66     document.addForm.name.value = "";
67     document.addForm.content.value = "";
68     alert("등록했습니다![" + comment.id + "]");
69 } else if (code == 'error') {
70     var message = xmlDoc.getElementsByTagName('message').item(0).firstChild.nodeValue;
71     alert("에러 발생:" + message);
72 } else {
73     alert("서버 에러 발생: " + req.status);
74 }
75 }
76 }
77 }
78
79 function updateComment() {
80     var id = document.updateForm.id.value;
81     var name = document.updateForm.name.value;
82     var content = document.updateForm.content.value;
83     var regdate = document.updateForm.regdate.value;
84     var params = "id=" + encodeURIComponent(id) + "&" +
85     "name=" + encodeURIComponent(name) + "&" +
86     "content=" + encodeURIComponent(content) + "&" +
87     "regdate = " + encodeURIComponent(regdate);
88     new ajax.xhr.Request('/board_comment/comment/update.do', params, updateResult, 'POST');
89 }
90
91 function updateResult(req) {
92     if (req.readyState == 4) {
93         if (req.status == 200) {
94             var xmlDoc = req.responseXML;
95             var code = xmlDoc.getElementsByTagName('code').item(0).firstChild.nodeValue;
96             if (code == 'success') {

```

```

97    hideUpdateForm() ;
98    var comment = eval("("
99    xmlDoc.getElementsByTagName('data').item(0).firstChild.nodeValue+ ")");
100   var listDiv = document.getElementById('commentList');
101   var newCommentDiv = makeCommentView(comment);
102   var oldCommentDiv = document.getElementById('c' + comment.id);
103   listDiv.replaceChild(newCommentDiv, oldCommentDiv);
104   alert ("수정성공");
105 } else if (code == 'error') {
106   var message = xmlDoc.getElementsByTagName('message').item(0).firstChild.nodeValue;
107   alert("에러발생 : " + message);
108 }
109 } else {
110   alert("서버 에러 발생 : " + req.status);
111 }
112 }
113 }
114
115
116 function viewUpdateForm(commentId) {
117   var commentDiv = document.getElementById('c' + commentId);
118   var updateFormDiv = document.getElementById('commentUpdate');
119   if (updateFormDiv.parentNode != commentDiv) {
120     updateFormDiv.parentNode.removeChild(updateFormDiv);
121     commentDiv.appendChild(updateFormDiv);
122   }
123   var comment = commentDiv.comment;
124   document.updateForm.id.value = comment.id;
125   document.updateForm.name.value = comment.name;
126   document.updateForm.content.value = comment.content;
127   document.updateForm.regdate.value = comment.regdate ;
128   alert (document.updateForm.regdate.value) ;
129   updateFormDiv.style.display = "";
130 }

```

```
131
132 function cancelUpdate() {
133     hideUpdateForm();
134 }
135
136 function hideUpdateForm() {
137     var updateFormDiv = document.getElementById('commentUpdate');
138     updateFormDiv.style.display = 'none';
139     updateFormDiv.parentNode.removeChild(updateFormDiv);
140     document.documentElement.appendChild(updateFormDiv);
141 }
142
143 function confirmDeletion(commentId) {
144     if(confirm("삭제하시겠습니까?")) {
145         var params = "id=" + commentId;
146         new ajax.xhr.Request('/board_comment/comment/delete.do', params, removeResult, 'POST');
147     }
148 }
149
150 function removeResult(req) {
151     if (req.readyState == 4) {
152         if (req.status == 200) {
153             var xmlDoc = req.responseXML;
154             var code = xmlDoc.getElementsByTagName('code').item(0).firstChild.nodeValue;
155             if (code == 'success') {
156                 var deleteId = xmlDoc.getElementsByTagName('id').item(0).firstChild.nodeValue;
157                 var commentDiv = document.getElementById("c" + deleteId);
158                 commentDiv.parentNode.removeChild(commentDiv);
159                 alert("삭제 했습니다.");
160             } else if (code == 'error') {
161                 var message = xmlDoc.getElementsByTagName ('message').item(0).firstChild.nodeValue;
162                 alert("에러발생 :" + message);
163             }
164         } else {
```

```

165     alert("서버 에러 발생 :" + req.status);
167 }
168 }
169 }

```

3.13.3list.jsp

WebContent/comment/list.jsp

```

01 <?xml version="1.0" encoding="UTF-8"?>
02 <%@ page contentType="text/xml; charset=UTF-8"%>
03 <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
04 <%@ taglib prefix="fmt" uri=" http://java.sun.com/jsp/jstl/fmt"%>
05
06 <result> <code>success</code> <data><![CDATA[
07 [
08 <c:forEach items="${list}" var="list" varStatus="i">
09   <c:if test="${i.count > 1}"> , </c:if>
10   {
11     id: ${list.comment_id},
12     num: ${list.num},
13     name: '${list.comment_name}',
14     content: '${list.comment}',
15     regdate: '<fmt:formatDate
16           value = "${list.regdate }" type = "DATE"
17           pattern="yyyy-MM-dd HH:mm:ss" /> '
18   }
19 </c:forEach>
20 ]
21 ]]> </data> </result>

```

3.13.4write.jsp

WebContent/comment/write.jsp

```

01 <?xml version="1.0" encoding="UTF-8" ?>
02 <%@ page contentType="text/xml; charset=UTF-8"%>

```

```

03 <%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>
04 <result> <code>success</code> <data><![CDATA[
05 {
06 id: ${comment.comment_id},
07 num: ${comment.num},
08 name: '${comment.comment_name}',
09 content: '${comment.comment}',
10 regdate: '<fmtformatDate value = "${comment.regdate}"
11 type = "DATE" pattern = "yyyy-MM-dd HH:mm:ss" />'
12 ]]> </data> </result>

```

3.13.5update.jsp

WebContent/comment/update.jsp

```

01 <?xml version="1.0" encoding="UTF-8"?>
02 <%@ page contentType="text/xml; charset=UTF-8"%>
03 <%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>
04
05 <result> <code>success</code> <data><![CDATA[
06 {
07 id: ${comment.commen_id},
08 num: ${comment.num},
09 name: '${comment.comment_name}',
10 content: '${comment.comment}',
11 regdate: '<fmtformatDate value = "${comment.regdate}"
12 type = "DATE" pattern = "yyyy-MM-dd HH:mm:ss" />' 
13 }
14 ]]> </data> </result>

```

3.13.6list.jsp

WebContent/comment/delete.jsp

```

01 <?xml version="1.0" encoding="UTF-8"?>
02 <%@ page contentType="text/xml; charset=UTF-8"%>
03 <result>

```

04	<code>success</code>
05	<id>\${id}</id>
06	</result>