프로젝트로 배우는

자바 웹 프로그래밍

Servlet, JSP, JDBC

황희정 지음

Chapter 14. 웹 애플리케이션 로그와 오류 관리

한빛아카데미 Hanbit Academy, Inc.

목차

1. 로그 관리 개요
2. [기본실습]SLF4J와 Log4j를 연동한 로깅 구현
3. 웹 애플리케이션 로깅 기법
4. [응용실습]시스템 전반에 걸친 로깅 정책 구현
5. 웹 애플리케이션 오류 관리
6. [기본실습]서버 기반 오류 처리

학습목표

- 웹 애플리케이션 로그의 필요성을 이해한다.
- 로그의 분류 및 구현 방안을 학습한다.
- SLF4J와 Log4J를 배우고 프로젝트에 활용한다.
- 서버에서 발생하는 오류에 대해 효과적으로 처리하는 방법을 배운다.

01. 로그 관리 개요

1. 로그 관리의 필요성

- 로그(Log)는 프로그램을 개발하거나 운영할 때 발생하는 문제점을 추적하거나 운영 상태를 모니터링하기 위한 텍스트 형식의 데이터를 말한다.
- 콘솔에 출력되는 메시지를 기본으로 파일 및 데이터베이스 연계 등 시스템 규모가 커질수록 더욱 체계적인 로그 관리가 요구된다.
- 일반적으로 로깅은 기록을 남긴다는 의미이므로, 단순한 메시지의 출력보다는 파일 형태의 기록으로 남기는 형태를 말한다. 로깅의 필요성을 정리하면 다음과 같다.

- 프로그램 개발 과정에서의 간단한 구현 검증
- 운영 시스템에서 프로그램 진행 과정의 모니터링
- 서버 기반 프로그램에서 사용자 접속 및 시스템 이용 관리
- 문제 발생 시 원인 파악을 위한 시스템 분석
- 사용자 접속 및 요청 등 보안 관점에서의 데이터 관리
- 이용 형태 분석을 통한 사후 서비스 개선 목적

01. 로그 관리 개요

2. 로그 유형 및 로깅 구현 형태

- 로그는 크게 애플리케이션 로그와 시스템 로그로 구분할 수 있다.

[표 14-1] 로그 유형

유형	설명
애플리케이션 로그	애플리케이션 내부 동작 과정에서 발생하는 내용에 대한 로깅 예 • 네트워크 문제로 인한 데이터베이스 연결 실패 • 데이터베이스 내부 처리에서 발생한 문제 • 서버 시작/종료 관련 메시지 • 기타 서버 프로그램 동작 중 발생한 문제 등
시스템 로그	사용자나 데이터와 관련된 내용에 대한 로깅 예 • 사용자 로그인 히스토리 • 웹의 경우 특정 페이지 요청 기록 • 사용자 권한 관련 • 특정 메서드 호출 및 매개변수 정보 • 처리 데이터 확인용 메시지 등

01. 로그 관리 개요

- 애플리케이션이나 시스템 로깅을 적용하기 위해서는 로깅 전용 프레임워크가 요구된다.
- 로깅 프레임워크를 사용하면 보다 체계적이고 편리하게 로그 메시지에 대한 유지보수가 가능하다.

[표 14-2] 대표적인 로깅 프레임워크

형태	설명
프로그래머가 임의로 구현한 로깅	프로그래머가 임의로 로깅 정책을 수립하고, 파일 입출력 기반의 로깅 클래스 구현 등을 통한 처리 방법이다. 비교적 손쉽게 구현할 수 있으나, 표준화가 되지 않았기 때문에 유지보수에 문제가 있을 수 있다.
JCL(Jakarta Commons Logging)을 이용한 로깅	웹 애플리케이션뿐만 아니라 일반 애플리케이션까지 범용으로 이용할 수 있는 로깅 프레임워크다. 여러 로깅 API들을 단일화된 접근 방식으로 사용할 수 있도록 해주는 것으로, 보통의 경우 내부적으로는 Log4j, JDK, Avalon 등의 로깅 API를 사용하면서 애플리케이션에서는 JCL을 이용해 프로그래밍하는 형태다.
Log4J를 이용한 로깅	Log4J는 가장 대표적인 로깅 프레임워크로 알려져 있다. 비교적 쉽게 이용할 수 있으며 다양한 기능을 제공한다. Log4j 단독으로도 사용될 수 있지만, commons-logging, SLF4J 등과 연계해서 많이 사용한다.
SLF4J(Simple Logging Framework)를 이용한 로깅	가장 최근에 나온 로깅 프레임워크다. 기존 로깅 API들의 문제점들을 보완한 솔루션으로 기존 로깅 프레임워크와의 호환성과 함께 성능적인 측면의 장점으로 인해 최근의 프로젝트들에서는 SLF4J를 많이 사용한다.

01. 로그 관리 개요

- 웹 애플리케이션에서는 서블릿 컨테이너가 제공하는 로깅 API를 사용할 수 있으며 필터나 리스너를 통해 시스템 관리에 필요한 로그 메시지를 처리할 수 있다.
- 또한 Aspect Oriented Programming 기법을 이용해 전체 애플리케이션에 일관된 로깅 구현을 하는 것도 가능하다.

[표 14-3] 웹 애플리케이션 로깅 구현 형태

형태	설명
서블릿 컨테이너를 이용한 로깅	웹 애플리케이션에서 대표적으로 이용할 수 있는 로깅의 구현 형태다. 톰캣과 같은 컨테이너에서 제공하는 API를 이용해 로깅을 하는 방식으로 톰캣 로그 파일을 이용하게 된다. 시스템 로깅의 처리에는 적합하지만, 애플리케이션 로깅을 처리하기에는 관리적인 문제가 있다.
필터 혹은 리스너를 이용한 로깅	웹 애플리케이션의 특징적인 로깅 구현 형태다. 필터의 특성상 적용 대상을 선택할 수 있으며 request, response 정보를 가로챌 수 있기 때문에 기존 애플리케이션을 수정하지 않고 추가적으로 필요한 로깅을 수행할 수 있다. 리스너의 경우에도 리스너 특징을 이용해 웹 애플리케이션 전반에 걸쳐 필요한 로깅을 수행할 수 있다.
AOP를 이용한 로깅	AOP는 Aspect Oriented Programming의 약자로, 로깅에 국한된 것은 아니며 현재 운영 중인 애플리케이션의 코드를 수정하지 않고 기능을 수정하거나 추가할 수 있는 새로운 프로그래밍 기법이다. 로깅 자체가 정형화된 코드 형태와 구조이므로 전체 코드에 대해 체계적인 접근으로 생각해볼 만한 방법이며 스프링 프레임워크 등을 사용하면 더욱 쉽게 이용할 수 있다.

01. 로그 관리 개요

- 로그 메시지는 메시지의 중요도나 성격에 따라 다음과 같은 로그레벨을 가진다.
- 일반적으로 로깅은 지정한 로그레벨 보다 상위 수준의 레벨을 출력한다.
- 로그 레벨을 두는 이유는 일괄적인 로그 정책을 적용하거나 관리 차원에서의 메시지 구분을 위해서이다.

[표 14-4] 로그 처리 메서드 및 로그 레벨

레벨	처리 메서드
FATAL	가장 심각한 오류를 말한다. 상태 콘솔에 즉시 출력한다.
ERROR	일반적인 오류를 말한다. 상태 콘솔에 즉시 출력한다.
WARN	오류는 아니지만 주의를 요구하는 경우를 말한다. 더 이상 쓸모없는 API 사용 등이 여기에 해당한다. 상태 콘솔에 즉시 출력한다.
INFO	런타임 시 관심 있는 이벤트를 말한다. 상태 콘솔에 즉시 출력한다.
DEBUG	시스템 흐름과 관련된 상세 정보를 말한다. 로그 파일로만 출력한다.
TRACE	가장 상세한 형태의 정보를 말한다. 로그 파일로만 출력한다.

3. SLF4J

- SLF4J는 Simple Logging Framework의 자바 버전으로, 애플리케이션에서 로깅 객체를 사용할 수 있는 기본적인 구조만을 제공하고 실제 로깅 처리는 다른 구현체를 쓸 수 있도록 해주는 범용 프레임워크다.
- SLF4J 는 기본적으로 라이브러리 형태로 제공되고 실제 로깅 처리를 위해 필요한 jar 파일을 추가적으로 프로젝트 빌드 경로에 추가해주어야 한다.
- 기본 API파일인 slf4j-api-1.7.5.jar은 반드시 필요하고, 실제 로깅 구현체로 사용할 slf4j-simple-1.7.5.jar 혹은 slf4j-log4j12-1.7.5.jar 등의 라이브러리는 필요에 따라 추가로 설치해야 한다.
- slf4j-simple-1.7.5.jar은 간단한 콘솔 출력에 필요한 것이고, slf4j-log4j-1.7.5.jar은 Log4j 연동에 필요한 것이다.
- 로깅 라이브러리는 [WEB-INF/lib] 폴더에 복사해 두어야 한다.
  - **SLF4J 라이브러리** : slf4j-api-1.7.5.jar
  - **로깅 구현체 중 택일(반드시 둘 중 하나만 복사되어 있어야 함)** : slf4j-simple-1.7.5.jar

    혹은 slf4j-log4j12-1.7.5.jar
  - **Log4J 라이브러리** : log4j-1.2.17.jar

01. 로그 관리 개요

- SLF4J 를 애플리케이션에서 사용하기 위해서는 다음과 같이 세가지 절차가 필요하다.

로깅 API에 대한 패키지 불러오기

```
import org.slf4j.Logger
import org.slf4j.LoggerFactory;
```

로깅을 위한 Logger 객체 확보

```
Logger logger = LoggerFactory.getLogger(Class clazz);
Logger logger = LoggerFactory.getLogger(String name);
```

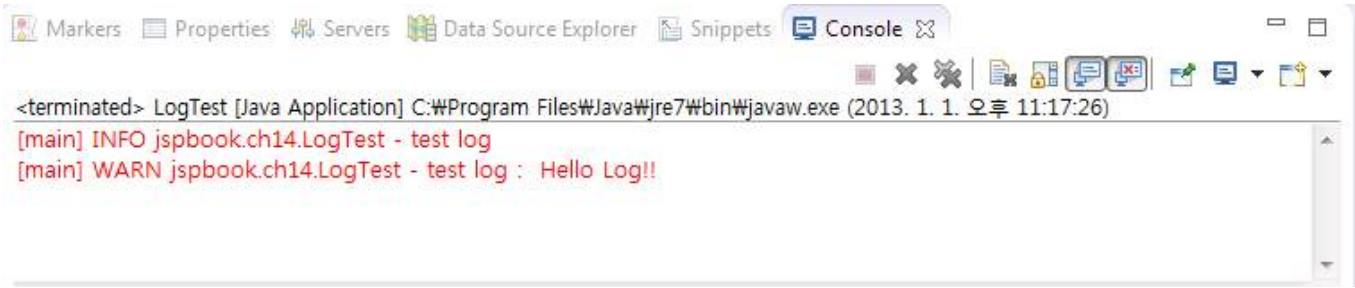
로깅 처리

```
log.info(Object message);
log.debug("Alert msg is {}", msg);
```

01. 로그 관리 개요

- 간단한 예제를 통해 SLF4J 의 사용법을 실습해 보자.
- **LogTest.java** – 교재 p.589 ~ 590 참고
  - 먼저 로거 객체를 확보하고 로그 레벨을 지정해 메시지를 출력한다.
  - 로그 메시지는 콘솔에 출력 된다.
  - 단순히 System.out.println()을 이용하는 것과 달리 메서드, 클래스, 로그메시지 등 여러 정보가 체계적으로 출력되는 것을 볼 수 있다. 메시지 포맷은 별도 설정을 통해 변경이 가능하다.

```
06 public class LogTest {
07
08     public static void main(String[] args) {
09         String msg = "Hello Log!!";
10         Logger logger = LoggerFactory.getLogger(LogTest.class);
11         logger.info("test log");
12         logger.warn("test log : {}",msg);
13     }
```



[그림 14-1] LogTest 실행결과

01. 로그 관리 개요

4. Log4J

- Log4j는 현재 가장 널리 사용되고 있는 오픈소스 로깅 프레임워크로, 다양한 설정을 통해 체계적인 로그 관리를 지원한다.
- Log4j는 자체 API를 제공하기 때문에 해당 API를 이용해 프로그래밍할 수 있지만, 호환성을 유지하려면 가급적 SLF4J API를 이용해 프로그래밍할 것을 권한다.

Log4j 설치하기

- Log4j 는 자바 이외에도 다음과 같이 다양한 언어를 지원한다.
  - Log4Cxx(C++)
  - Log4j(1.x, 2-beta)
  - Log4Net
  - Log4PHP

- Log4j 역시 라이브러리 파일을 [WEB-INF/lib] 폴더에 복사해야 사용이 가능하다.

01. 로그 관리 개요

Log4j 설정하기

- Log4j를 사용하려면 먼저 log4j 설정 파일을 만들어야 한다. log4j는 단순히 로그를 출력하는 역할만 하는 것이 아니라 로그 메시지의 형식, 저장 방식 등을 다양하게 지정할 수 있다.
- 설정방식은 두 가지로, 자바의 속성 파일 혹은 xml 형식을 선택해 사용할 수 있다. xml 설정 방식이 나중에 나왔고 좀 복잡하기는 하지만 더욱 구조적이고 프로퍼티 파일보다 다양하게 설정할 수 있다.
- 설정 파일은 자바의 클래스패스 경로 상에 있으면 되고 이클립스에서 개발하는 웹 애플리케이션의 경우 src 폴더에 넣어두면 된다.

[표 14-5] log4j.xml 주요 요소

요소	내용
〈root〉 〈/root〉	root 로거를 정의한다. 기본 로그 설정이라고 보면 된다. • level : FATAL, ERROR, WARN, INFO, DEBUG • appender-ref : 해당 appender의 세부 설정을 할 때 사용할 이름
〈logger〉 〈/logger〉	관리할 로거를 정의한다. 패키지별 로그 레벨에 따른 로거 이름을 설정한다. • level : FATAL, ERROR, WARN, INFO, DEBUG • appender-ref : 해당 appender의 세부 설정을 할 때 사용할 이름
〈appender〉 〈/appender〉	root에서 설정한 appender 이름에 대한 실제적인 appender와 레이아웃 등을 지정한다. • appender 클래스 종류 참조([표 14-6]) • layout 클래스 종류 참조([표 14-7])

01. 로그 관리 개요

- **Logger**
  - Logger는 Log4j 에서 로그 설정을 그룹화한 것으로 이해하면 된다.
  - 기본적으로 root 로거가 있고 그 외에는 추가적으로 로거를 만들어 정의하면 된다.
- **appender**
  - appender는 로그 메시지의 운영 및 관리 유형을 지정하는 것으로, 정해진 appender에 따라 로그 파일의 형태나 구조가 달라지게 된다.

[표 14-6] appender 주요 클래스 종류(기본 패키지 : org.apache.log4j.XXX)

종류	내용
ConsoleAppender	콘솔에 로그 메시지를 출력한다.
FileAppender	파일에 로그 메시지를 기록한다.
RollingFileAppender	파일에 로그 메시지를 기록하고, 파일 크기가 일정 수준 이상이 되면 다른 이름으로 저장하고 새롭게 기록한다.
DailyRollingFileAppender	파일에 로그 메시지를 기록하고, 하루 단위로 로그 파일을 변경해서 저장한다.
SMTPAppender	로그 메시지를 이메일로 전송한다.
NTEventLogAppender	NT 시스템 이벤트 로그로 메시지를 전송한다. 윈도우 NT/XP/2003 서버 등에 해당한다.

01. 로그 관리 개요

- **layout**
  - layout은 appender에서 사용할 로그 메시지의 포맷을 지정하는 것이다.
  - [표14-7]과 같은 기본 layout이 있고 보통은 PatternLayout을 이용해 메시지 형식을 정의해 사용한다.

[표 14-7] layout 클래스 종류(기본 패키지 : org.apache.log4j.Layout)

종류	내용
DateLayout	로그 메시지를 날짜 중심으로 간단하게 기록한다.
HTMLLayout	로그 메시지를 HTML 형식으로 기록한다.
PatternLayout	로그 메시지를 사용자 지정 패턴에 따라 기록한다.
SimpleLayout	레벨-메시지 형식의 가장 간단한 로그를 기록한다.
XMLLayout	로그 메시지를 XML 형식으로 기록한다.

01. 로그 관리 개요

- PatternLayout의 메시지 형식은 교재 593 ~ 594페이지의 [표 14 - 8]을 참조 한다.

%p debug, info, warn, error, fatal 등의 priority 가 출력된다.
%m 로그내용이 출력됩니다
%d 로깅 이벤트가 발생한 시간을 기록합니다.
  포맷은 %d{HH:mm:ss, SSS}, %d{yyyy MMM dd HH:mm:ss, SSS}같은 형태로
  사용하 며 SimpleDateFormat에 따른 포맷팅을 하면 된다
%t 로그이벤트가 발생된 쓰레드의 이름을 출력합니다.
%% % 표시를 출력하기 위해 사용한다.
%n 플랫폼 종속적인 개행문자가 출력된다. rn 또는 n 일것이다.
%c 카테고리를 표시합니다
  예) 카테고리가 a.b.c 처럼 되어있다면 %c{2}는 b.c가 출력됩니다.
%C 클래스명을 포시합니다.
  예)클래스구조가 org.apache.xyz.SomeClass 처럼 되어있다면 %C{2}는
      xyz.SomeClass 가 출력됩니다
%F 로깅이 발생한 프로그램 파일명을 나타냅니다.
%l 로깅이 발생한 caller의 정보를 나타냅니다
%L 로깅이 발생한 caller의 라인수를 나타냅니다
%M 로깅이 발생한 method 이름을 나타냅니다.
%r 어플리케이션 시작 이후 부터 로깅이 발생한 시점의 시간(milliseconds)
%x 로깅이 발생한 thread와 관련된 NDC(nested diagnostic context)를 출력합니다.
%X 로깅이 발생한 thread와 관련된 MDC(mapped diagnostic context)를 출력합니다.

02. [기본실습]SLF4J와 Log4j를 연동한 로깅 구현

원리를 알면 IT가 맛있다
IT COOKBOOK

1. 실습 개요

- 간단한 설정을 통해 SLF4J와 Log4j를 연동한 로깅 예제를 살펴보자.
- 실습은 xml 형식의 log4j 설정을 사용하고, 설정 내용은 INFO 레벨의 경우 콘솔로 출력하고, WARN 메시지의 경우 파일에 로그 내용을 기록하는 형식으로 구성한다.
- 로깅 테스트는 앞서 SLF4J 테스트에서 사용한 LogTest.java를 그대로 이용한다.

[표 14-9] 실습 파일

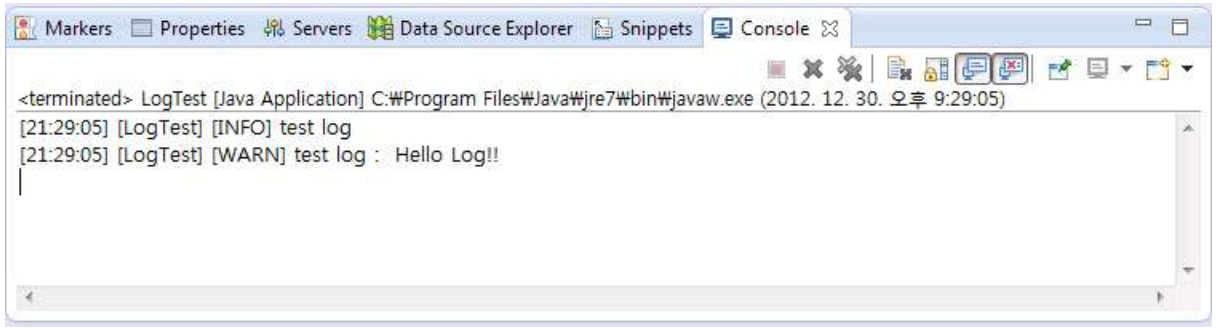
파일 이름	설치 위치
log4j.xml	src
LogTest.java	src₩jspbook₩ch14

02. [기본실습]SLF4J와 Log4j를 연동한 로깅 구현

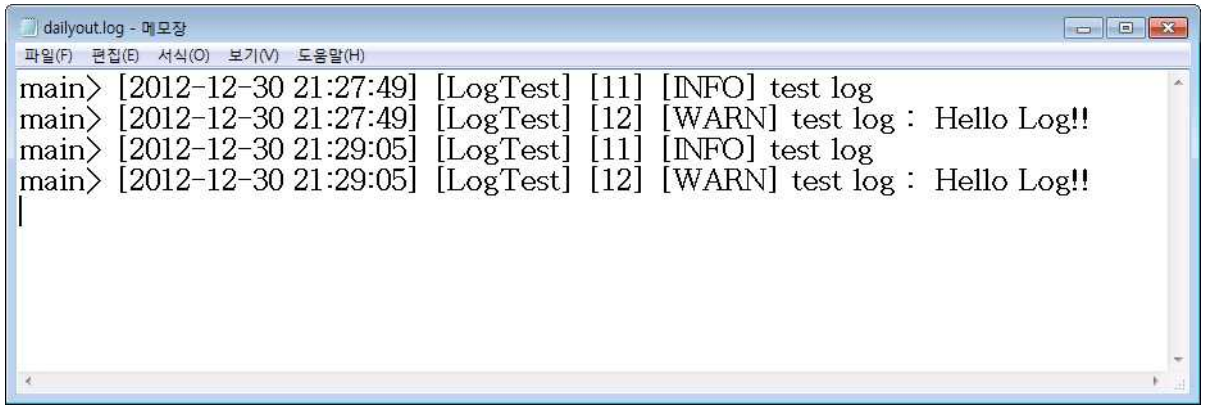
2. log4j.xml 작성

- **log4j.xml** – 교재 p.595 ~ 596 참고

3. 실행 결과 확인



[그림 14-2] LogTest 실행결과–콘솔



[그림 14-3] LogTest 실행결과–로그파일

03. 웹 애플리케이션 로깅 기법

1. 서블릿 컨테이너 제공 로깅

- 웹 애플리케이션 로깅은 jsp 등 웹 프로그램에서 기본적으로 사용할 수 있는 로깅 방법으로, 주로 시스템 로깅을 처리하기 위해 이용한다.
- 웹 애플리케이션의 경우 컨테이너와 밀접한 관계가 있으므로 서블릿 컨테이너 제공 로깅을 잘 알고 있으면 유용하게 활용할 수 있다.
- 컨테이너마다 로그 파일 이름이나 위치 등이 다르며, 여기에서는 톰캣을 기준으로 설명한다.

[표 14-10] ServletContext 클래스 로깅 메서드

메서드	설명
public void log(String msg)	컨테이너 로깅 시스템을 통해 문자열 형태의 내용을 출력한다. 기본 톰캣의 경우 localhost.yyyy-MM-dd.log 형태의 파일에 기록된다. 이클립스의 경우 콘솔 메시지로만 출력된다. 로그 레벨은 INFO로 취급된다.
public void log (String msg, Throwable throwable)	Throwable 클래스는 모든 자바 오류와 예외의 슈퍼 클래스다. 메시지와 함께 특정 예외 처리 메시지를 함께 로깅할 수 있다. 로그 레벨은 SEVERE(심각)로 취급된다.

03. 웹 애플리케이션 로깅 기법

2. 필터를 이용한 로깅

- 필터는 자바 웹 애플리케이션에서 구현 가능한 특수한 형태의 프로그램으로 특정 요청을 처리하기 이전에 실행될 수 있으므로 관리를 위한 별도의 로깅을 필터에서 구현하면 모든 jsp 실행 전에 필요한 로깅을 일괄적으로 처리할 수 있어 편리하다.
- 실제 로깅 구현은 SLF4J와 Log4j 를 사용하게 된다.

[표 14-12] 필터를 이용한 로깅 구현 시 고려사항

항목	설명
로그 파일 이름	보통 web.xml에 초기화 매개변수로 파일 이름을 등록한 후 참조한다.
로그 처리	init() 메서드 및 destroy() 메서드에서 처리한다. 로깅 프레임워크를 이용한다.
필터 매핑	필터의 특징은 특정 URL 패턴에 따라 여러 필터를 선별적으로 적용할 수 있다는 점이다. 로깅 정책에 따른 적절한 매핑 설정이 중요하다.

3. 리스너를 이용한 로깅

- 필터와 마찬가지로 리스너 역시 표준 서블릿 스펙이므로, 모든 웹 애플리케이션에서 호환 가능한 로깅을 구현할 수 있다.
- 리스너의 경우 컨테이너의 특정 이벤트에 기반을 둔 로깅 구현에 적합하며, 필터와 동일하게 시스템 로그와 애플리케이션 로그의 중간적인 정보를 처리한다.
- 필터와 마찬가지로 전체 시스템 설계 관점에서 고려되어야 하는 것으로 이러한 로깅 정책 수립은 어느 정도의 시스템 관리 및 운영 경험이 요구 된다.

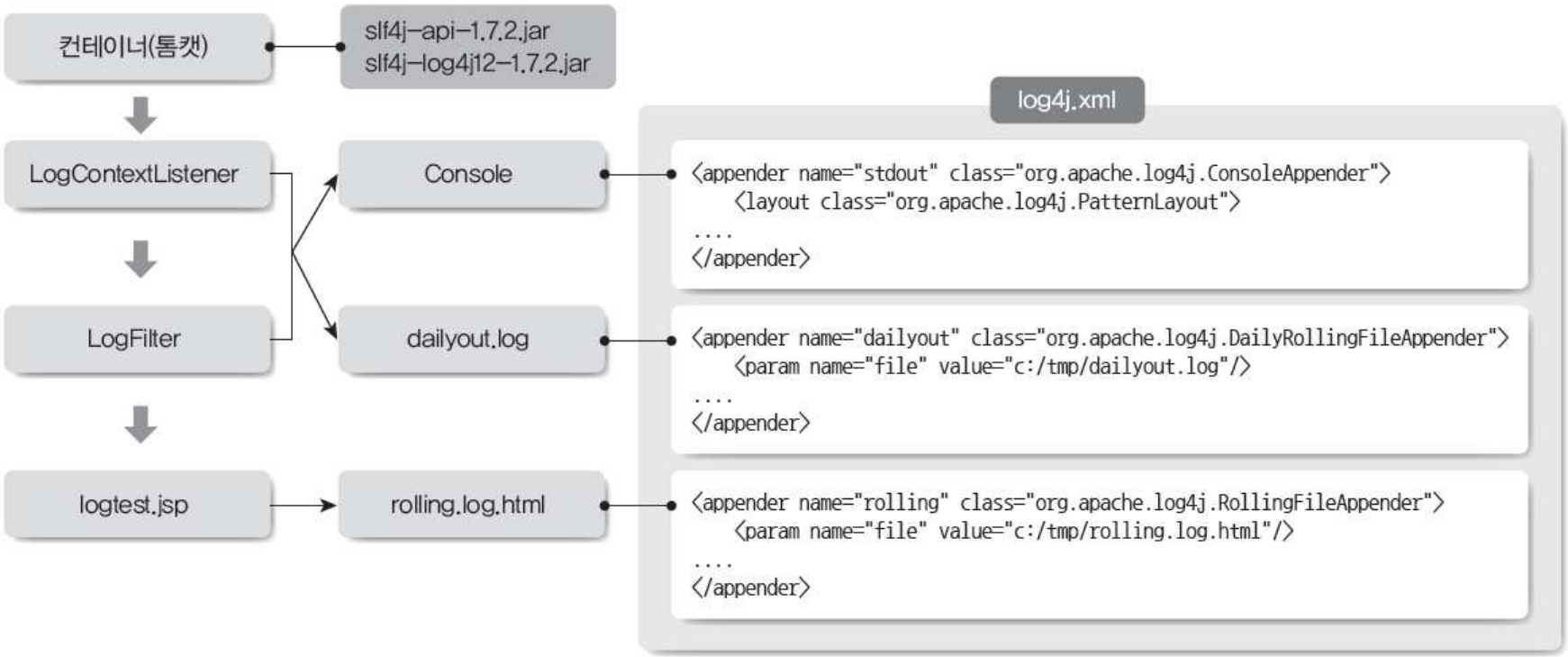
[표 14-13] 리스너를 이용한 로깅 구현 시 고려사항

항목	설명
로그 파일 이름	보통 web.xml에 초기화 매개변수로 파일 이름을 등록한 후 참조한다.
로그 처리	자카르타 Commons 및 Log4j 등의 로깅 프레임워크를 이용하며 다음과 같은 리스너 생명주기 메서드에서 처리한다. • contextInitialized() 및 contextDestroyed() 메서드 • sessionCreated() 및 sessionDestroyed() 메서드

04. [응용실습]시스템 전반에 걸친 로깅 정책 구현

1. 실습 개요

- 지금까지 웹 애플리케이션에서의 다양한 로깅 기법들을 살펴보았다. 이제부터는 지금까지의 내용을 정리하는 차원에서 jspbook 웹 애플리케이션에 대한 전반적인 로깅 정책을 수립하고, 필터, 리스너 등을 이용한 웹 애플리케이션 로깅 구현을 실습해보자.
- 예제는 다음과 같은 시나리오에 따라 여러 로깅 기법을 적용해 웹 애플리케이션 전체 운영에 필요한 로깅 시스템을 구축하게 된다.



[그림 14-4] 로깅 예제 시나리오

04. [응용실습]시스템 전반에 걸친 로깅 정책 구현

2. log4j.xml 파일 설정

- 앞에서 만들었던 log4j.xml에 몇가지 새로운 설정을 추가 함.
- root 로거 이외에 jspbook.ch14 패키지의 자바 클래스 및 ch14 폴더 아래의 jsp에 대한 로거를 지정함.
- rolling appender는 HTML Layout을 사용하며 파일크기가 10kb가 되면 새로 생성한다.
- **log4j.xml** – 교재 p.602 ~ 603 참고

```
20  <appender name="rolling" class="org.apache.log4j.RollingFileAppender">
21      <param name="file" value="c:/tmp/rolling.log.html"/>
22      <param name="Append" value="true"/>
23      <param name="MaxFileSize" value="10kb"/>
24      <param name="MaxBackupIndex" value="1"/>
25      <layout class="org.apache.log4j.PatternLayout" />
26  </appender>
```

- ch14 폴더의 jsp에서 발생하는 WARN 로그 메시지에 대해 rolling appender를 적용한다.

```
33  <logger name="org.apache.jsp.ch14">
34      <level value="WARN" />
35      <appender-ref ref="rolling" />
36  </logger>
```

04. [응용실습]시스템 전반에 걸친 로깅 정책 구현

원리를 알면 IT가 맛있다
IT COOKBOOK

3. 리스너 로그 구현

- **LogContextListener.java** – 교재 p.604 ~ 605 참고

4. 필터 로그 구현

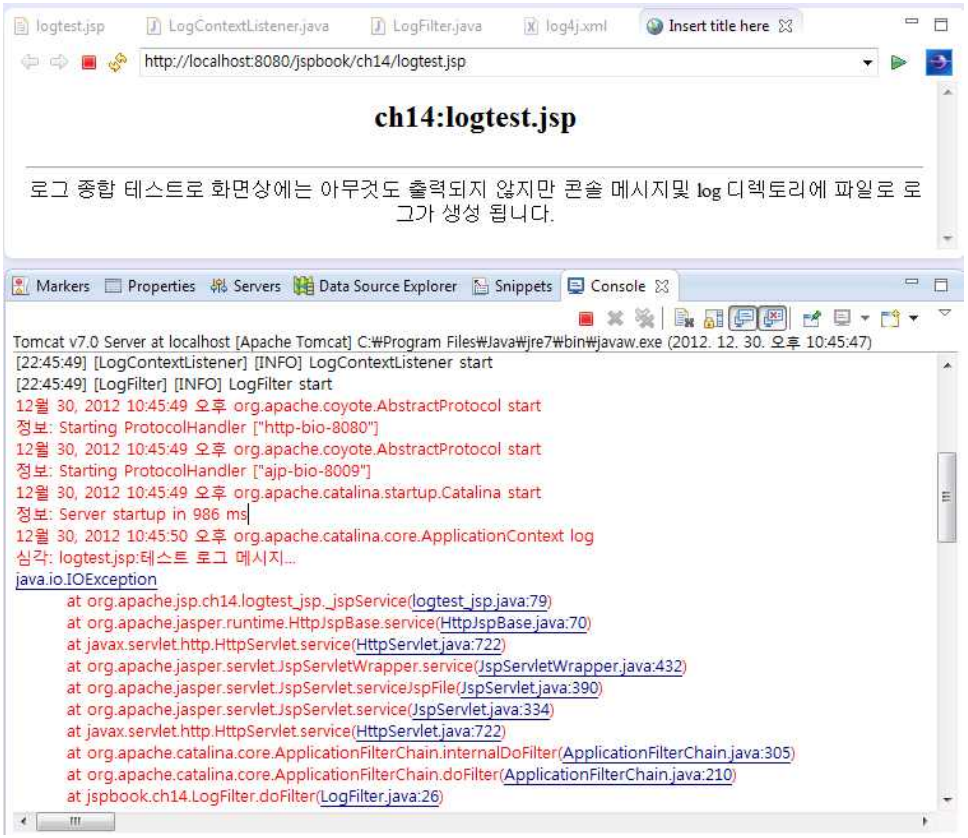
- **LogFilter.java** – 교재 p.605 ~ 606 참고

5. logtest.jsp 작성

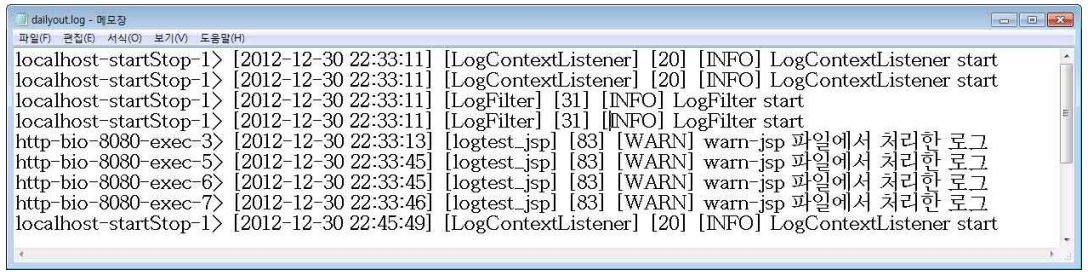
- **logtest.jsp** – 교재 p.607 참고

04. [응용실습]시스템 전반에 걸친 로깅 정책 구현

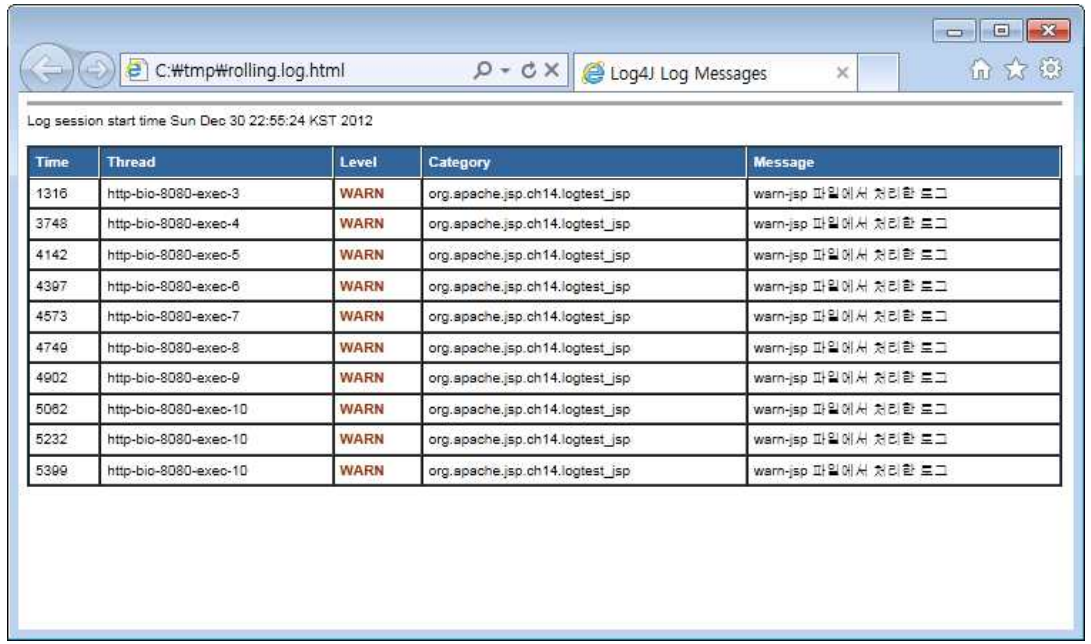
6. 실행 및 결과 확인



[그림 14-5] 콘솔 로그 메시지 결과



[그림 14-6] dailyout.log 파일 내용



[그림 14-7] rolling.log.html 파일 내용

05. 웹 애플리케이션 오류 관리

1. 컨테이너 오류 처리 설정

- 컨테이너 오류 처리 설정은 톰캣과 같은 애플리케이션 서버 설정을 통해 웹 애플리케이션 단위별로 오류 상황에 대처 하도록 하는 방법이다.
- 즉, 웹 애플리케이션 실행 중 발생하는 오류 유형별로 페이지를 지정해서 오류를 처리하는 방식이다.
- 웹 애플리케이션 실행 중 발생하는 오류는 컨테이너에서 출력하는 것으로 웹 표준 프로토콜인 HTTP 규격에 정의된 오류 코드에 따라 처리 된다.

[표 14-14] 대표적인 웹 애플리케이션 오류 처리 코드

오류 처리 코드	내용
100	Continue
200	OK, 오류 없이 전송 성공
201	Created, POST 명령 실행 및 성공
400	Bad request, 클라이언트의 잘못된 요청으로 처리할 수 없음
403	Forbidden, 접근이 거부된 문서를 요청함

05. 웹 애플리케이션 오류 관리

오류 처리 코드	내용
404	Not found, 문서를 찾을 수 없음
405	Method not allowed, 리소스를 허용하지 않음
406	Not acceptable, 허용할 수 없음
500	Internal server error, 내부 서버 오류(잘못된 스크립트 실행 시)
501	Not implemented, 클라이언트에서 서버가 수행할 수 없는 행동을 요구함
502	Bad gateway, 서버의 과부하 상태
503	Service unavailable, 외부 서비스가 죽었거나 현재 멈춘 상태
505	HTTP version not supported

05. 웹 애플리케이션 오류 관리

- 오류 코드에 따라 별도의 파일로 로그를 처리 하려면 web.xml에 다음과 같이 코드별 에러처리 파일을 지정해 주어야 한다.

[표 14-15] web.xml 오류 처리

오류 유형	사용 예
프로그램 오류	<error-page> <exception-type>java.lang.Throwable</exception-type> <location>/error/error.jsp</location> </error-page>
서버 오류	<error-page> <error-code>500</error-code> <location>/error/500-error.jsp</location> </error-page>

05. 웹 애플리케이션 오류 관리

2. JSP 오류 처리

- 이번에는 JSP에서 발생하는 오류 처리에 대해 살펴본다.
- JSP page 지시어 부분을 다룰 때 배운 errorPage와 isErrorPage 속성을 이용한 오류 처리를 살펴 보았음.
- errorData 객체를 사용해 좀 더 세밀한 에러 메시지 처리가 가능함.

[표 14-16] errorData 속성

속성	데이터 유형	설명
requestURI	String	요청이 실패한 URI
servletName	String	실패한 JSP나 서블릿 이름
statusCode	int	실패 상태 코드
throwable	Throwable	오류 페이지를 불러들인 예외

errorData 사용 예

```
요청 실패 URI : ${ pagecontext.errorData.requestURI }
상태 코드 : ${ pageContext.errorData.statusCode }
예외 유형 : ${ pageContext.errorData.throwable }
```

06. [기본실습]서버 기반 오류 처리

원리를 알면 IT가 맛있다
IT COOKBOOK

1. 실습 개요

- 실습을 통해 앞에서 배운 서버 오류 및 페이지 오류 처리의 실제 활용 방법을 배운다.
- 컨테이너 및 JSP 오류 처리를 모두 실습 한다.

2. web.xml 수정

- **오류 처리 추가 web.xml** – 교재 p.613 참고
- 서버 오류 처리를 위해 web.xml에 다음 내용을 추가한다.

```
01 <error-page>
02     <exception-type>java.lang.Throwable</exception-type>
03     <location>/error/error.jsp</location>
04 </error-page>
05 <error-page>
06     <error-code>404</error-code>
07     <location>/error/404-error.jsp</location>
08 </error-page>
```

06. [기본실습]서버 기반 오류 처리

원리를 알면 IT가 맛있다
IT COOKBOOK

3. 서버 오류 파일 만들기

- **404-error.jsp** – 교재 p.614 참고
  - 404 오류 발생시 보여줄 jsp 페이지
  - pageContext를 이용해 에러 메시지 처리

```
22 요청 실패 URI : ${ pageContext.errorData.requestURI }<BR>
23 상태 코드 : ${ pageContext.errorData.statusCode }<BR>
```

4. 페이지 오류 파일 만들기

- **error_test.jsp** – 교재 p.615 참고
- **error_ch14.jsp** – 교재 p.616 참고
  - jsp 페이지에서 에러 발생시 보여줄 jsp 페이지

```
22 요청 실패 URI : ${ pageContext.errorData.requestURI }<BR>
23 상태 코드 : ${ pageContext.errorData.statusCode }<BR>
24 예외 유형 : ${ pageContext.errorData.throwable }
```

  - 에러처리 jsp 파일의 page 지시어에 isErrorPage="true" 속성이 들어가야 함.

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

06. [기본실습]서버 기반 오류 처리

5. 공용 오류 처리 파일 만들기

- **error.jsp** – 교재 p.617 ~ 618 참고

6. 실행 및 결과 확인

- 총 세가지 유형의 에러 처리를 구현 했으므로 다음과 같이 단계별로 확인함.
  - ❶ 페이지별 오류 처리 : error_test.jsp 실행 → error_ch14.jsp를 통한 오류 처리 확인
  - ❷ 공용 페이지 오류 처리 : error_test2.jsp 실행 → error₩error.jsp를 통한 처리 확인
  - ❸ 웹 서버 오류 처리 : 잘못된 URL 요청 → error₩404-error.jsp를 통한 처리 확인

[그림 14-8] error_test.jsp 실행 결과

[그림 14-9] error_test2.jsp 실행 결과

[그림 14-10] 404-error.jsp 파일 확인

프로젝트로 배우는

자바 웹 프로그래밍

Servlet, JSP, JDBC

황희정 지음