Makale içeriği Java ile kullanılabilen ve süreci yürütmekte en çok kullanılan LOG alma tekniklerinden bahsedilmiştir.

1. GENEL BAKIŞ

1.1. LOG NEDİR?

LOG, bilgisayarda gerçekleşen her etkinliğin kayıt alına alınması demektir. Her Java uygulaması da belli bir noktada LOG alma (günlüğe kaydetme) işlemi gerektirir. Özellikle WEB uygulamalarında günlüğe kaydetme işlemi çok önemli bir yere sahiptir. Hataları, uyarıları veya oluşabilecek her türlü krize karşı önceden oluşturulmuş kayıtlar sayesinde daha hızlı ve daha rahat erişim sağlayıp soruna çözüm üretmemizde yardımcı olur. Uygulamanın aksamamasını, yazılımcı ekibinin işini kolaylaştırmasını, hataların kaydedilip raporlanmasını ve analiz edilmesini sağlar.

1.2. LOG SEVİYELERİ

Günlük seviyeleri , bir mesajın ne kadar önemli olduğunu ve hangi mesajların yapacağını belirler. En çok kullanılan framework olan LOG4J için seviyeler şu şekildedir:

- **OFF:** Günlüğe kaydetme işlemini kapatmaya yöneliktir.
- **FATAL:** Acil durum mesajıdır. Örneğin SQL Serverin disk alanı dolmuş olsun, kullanıcılar veri kaydedemediğinden uygulama duracaktır. En önemli seviyedir.
- **ERROR:** Ele alınamayan(handled edilemeyen) hata mesajıdır. Uygulama akışı bozulmuştur. Örnek olarak bozulan dosya var ve sunucuya gönderilemiyor.
- **WARN:** Uygulama akışını bozmayan olaylar için kullanırız. Projenin ilerisi için sıkıntı oluşturabileceğinden bakılması önemlidir.
- **INFO:** Bilgi mesajıdır. Uygulamanın genel akışını takip etmek için kullanılır. Örnek olarak: istek alındı istek gönderildi gibi bilgiler yer alabilir.
- **DEBUG:** Hata ayıklama mesajıdır.
- **TRACE:** Debug için değerli bilgileri taşır. Uygulama açıldı, kapatıldı gibi.
- **ALL:** Tüm seviyeleri içerir.

Bazı log seviyeleri başka frameworkler'de farklılık gösterebilir.

1.3.**HANDLER**

Log level'den gelen mesajları herhangi bir yere yazdırıp, kaydetmeye yarar.

- **ConsoleHandler:** Log mesajlarını konsola yazar
- **FileHandler:** Log mesajlarını dosyaya yazar.

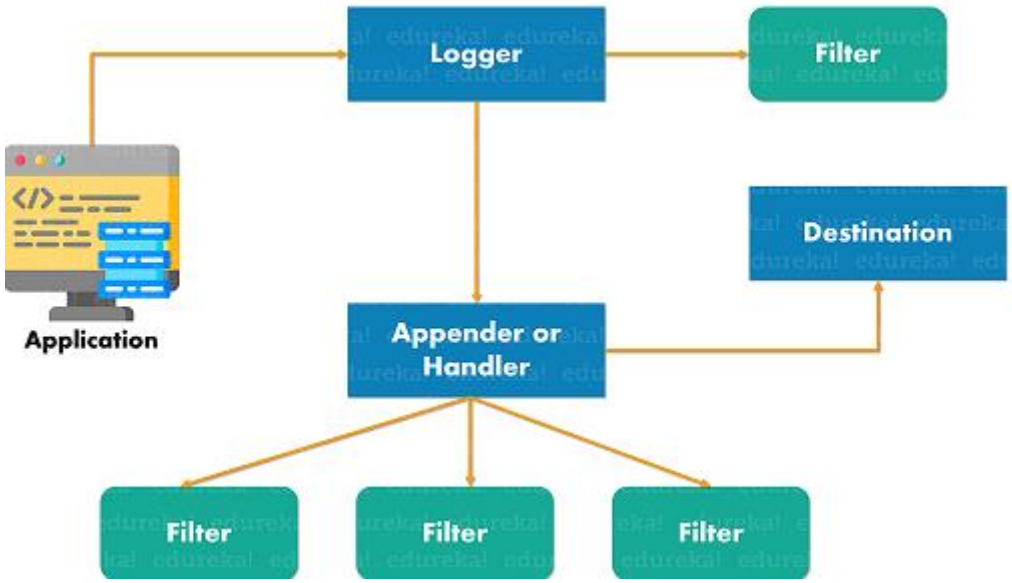
1.4.**FORMATTER:**

Yazılan ve kaydedilen mesajların daha okunaklı ve düzenli olması için gereklidir.

- **SimpleFormatter:** Metin formatındadır.
- **XMLFormatter:** XML formatındadır.

1.5.**LOG MANAGER:** Log'ların yaratılıp yapılandırılmasını ve yönetilmesini sağlar.

Java LOG işleminde temel bileşenler HANDLER, FORMATTER ve LOG MANAGER'dir.



Şekil 1: Temel LOG yapısı

Bir uygulama günlük araması yaptığında, Logger bileşeni filtrelemeden geçirerek ilgili Handler'a iletir. Ardından Formatter kullanarak kaydı gerekli formata göre biçimlendirir. Birden fazla formata çevirmek için birden fazla filtrelemeden geçirilir.

2. ÖRNEK FRAMEWORKLER

2.1. LOG4J

Burada LOG4J frameworku hakkında konuşacağız. APACHE firması tarafından sağlanan açık kaynaklı bir frameworktür. LOG4J kullanabilmek için iki yol vardır. İlki log4j.properties ikincisi ise log4j.xml dir. Bunlar ne tür bir günlük kaydı oluşturmak istediğimizle ilgili yapılandırmayı içeren dosyalardır. Eğer iki dosyayı da kullanmak istersek önceliği XML dosyasına verecektir. Bir log4j.properties dosyası ile oluşturalım.

İlk önce https://logging.apache.org/log4j/1.2/download.html üzerinden gerekli kütüphaneyi indirelim. Daha sonra INTELIJ üzerinden boş bir java projesi oluşturalım. File aracından Project Structure gelerek Modules kısmından indirdiğimiz zip dosyası içinde yer alan jar dosyasını seçelim. Bu jar dosyası External Libraries kısmında oluşacaktır.

İkinci adım olarak projemiz içindeki src klasörüne sağ tıklayıp 'log4j.properties' dosyası oluşturalım. Burası gelen log kayıtlarını nereye ve nasıl tutulacağını gösteriyor. Biz hem dosyaya hem de konsola kayıt edelim. https://www.javatpoint.com/log4j-properties bu link ile gerekli kodları alıp dosya içine yapıştıralım. Kod içinde yer alan dosya yolunu değiştirebilirsiniz. Ben C:\logs.log şeklinde yaptım.

Şimdi gelelim kod yazmaya.

```
import org.apache.log4j.Logger;
public class Main {

  private static Logger logger = Logger.getLogger(Main.class);

  // Logger ifadesini yazınca org.apache.log4j.Logger; import edilmeli.

    public static void main(String[] args) {

        logger.info("Info Message");
        logger.warn("Warn Message");
    }
}
```

Çıktısı log dosyasında şu şekildedir:

```
2022-07-05 15:42:13 INFO  Main:5 - Info Message
2022-07-05 15:42:13 WARN  Main:6 - Warn Message
```

2.2 JAVA UTİL LOGGİNG

Bu log düzeyi için dışarıdan dosya yükleme ihtiyacı yoktur. Kodun daha düzenli olması açısından log adında class oluşturmak işimizi rahatlatır. Burada log seviyeleri küçükten büyüğe doğru FINEST, FINER, FINE, CONFIG, INFO, WARNING, SEVERE şeklindedir. Ek olarak günlüğe kaydetmeyi kapatmak için OFF, tüm mesaj günlüğüne erişebilmek için ALL düzeyi kullanılabilir.

```
3  public class Log {
4
       public Logger logger;
       FileHandler fh;
5
       public Log(String file_name) throws SecurityException, IOException {

           File f = new File(file_name);
           if(!f.exists()){
               f.createNewFile();
           }
           fh = new FileHandler(file_name,true);
           logger = Logger.getLogger("test");
           logger.addHandler(fh);
           SimpleFormatter formatter = new SimpleFormatter();
           fh.setFormatter(formatter);
       }
   }
```

Burada dosya formatı ve eğer dosya yoksa otomatik oluşturulmasını istedik. “FileHandler” ile log dosyaları kaydetmeyi, “SimpleFormater” ile metin formatında çıktı almayı istedik.

```
public class Main {

    public static void main(String[] args) throws IOException {
        Log my_log = new Log("log.text");

        try {
            my_log.logger.setLevel(Level.INFO);
            my_log.logger.info("info message");
            my_log.logger.severe("severe message");
        }
        catch (Exception e){
            my_log.logger.log(Level.WARNING,"Exception",e);
        }
    }
}
```

setLevel ile en düşük seviyede yazmak istediğimiz seviyeyi seçtik. Biz Level.INFO seçtiğimiz için INFO, WARNING ve SEVERE seviyelerine erişebiliriz.

2.3 SLF4J (Simple Logging Facade for Java)

Bu logun amacı programlayabileceğimiz bir arayüzdür. Bu nedenle ileride farklı bir log framework kullanmak istersek kodu tekrar yazmadan geçiş yapabilme imkanı sağlar.

Bu sefer ECLİPSE üzerinden Maven-Java projesi oluşturduk. Burada hazırda gelen pom.xml dosyası bizim için önemli. Pom.xml dosyası üzerinde oluşturacağımız dependency'ler ile logback , slf4j ve log4j arasında geçiş yapabileceğiz.

Group Id: org.slf4j , Artifact Id: slf4j-api Version: 1.7.5 yazarak ilgili dependency pom.xml dosyası içinde oluşur. Ancak burada sınıf içinde oluştuğumuz metotlar çalışsa da logger.info gibi bilgiler çalışmayacaktır. Bunun için ek olarak Group Id: org.slf4j, Artifact Id: slf4j-nop yazarak pom.xml içinde dependency oluşacaktır. Bu slf4j için hiyerarşiyi sağlıyor.

Ek olarak Group Id: ch.qos.logback , Artifact Id: logback-classic Version: 1.0.13 dependency ile logback framework'üne ya da

Group Id: org.slf4j , Artifact Id: slf4j-log4j12, Version: 1.0.13 dependency ile de log4j geçiş yapabiliriz. Bu bize kod silmeden yada kod tekrarı yapmadan hızlı bir şekilde frameworkler arası geçişi kolaylaştırıyor.

Not: Versiyonlar değişiklik gösterebilir.

Slf4j için pom.xml dosyasında oluşan dependency:

```
<dependency>
<groupId>org.slf4j</groupId>
<artifactId>slf4j-api</artifactId>
<version>1.7.5</version>
</dependency>
```

Logback için pom.xml dosyasında oluşan dependency:

```
<dependency>
<groupId>ch.qos.logback</groupId>
<artifactId>logback-classic</artifactId>
<version>1.0.13</version>
</dependency>
```

Bu makalede Java programında en çok kullanılan popüler log alma tekniklerinden bahsettik. Bazı framework'ler dışarıdan .jar uzantılı dosya alarak bazıları da API üzerinden import edilerek çalışmaktadır. Bu dosyaları konsol üzerinden, xml yada notepad dosyaları üzerinden gerekli bilgileri alıp uygulama için gerekli adımları atmamızda yardımcı olmasını sağladık.

Sonuç olarak her ne kadar hazır kütüphaneler ile süreç idame ettiriliyor olsa dahi iş yükünü programcı isteği doğrultusunda bir platform üzerinden kendi sınıflarını, fonksiyonlarını yazarak da kütüphane ve platformdan bağımsız bir şekilde sürece devam edebilmektedir.

KAYNAKÇA:

1. Şekil-1 https://www.edureka.co sitesinden alınmıştır.
2. https://docs.oracle.com/javase/7/docs/api/java/util/logging/Logger.html
3. https://logging.apache.org/log4j/2.x/