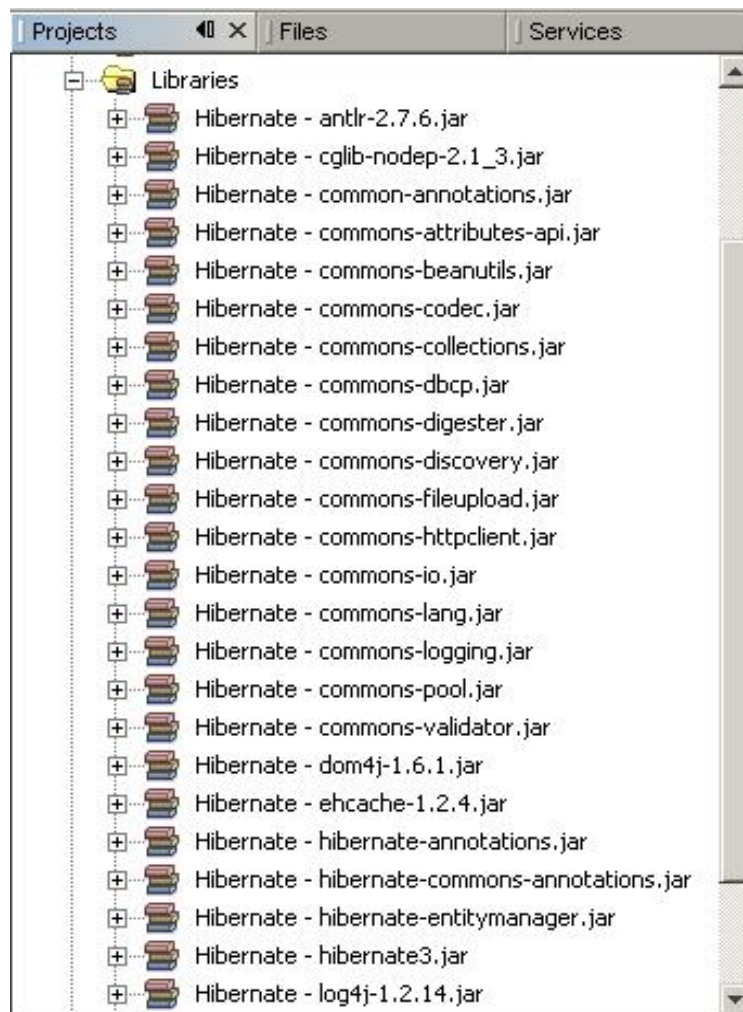


Proses CRUD Dengan Hibernate Annotations Menggunakan Netbean 6.0

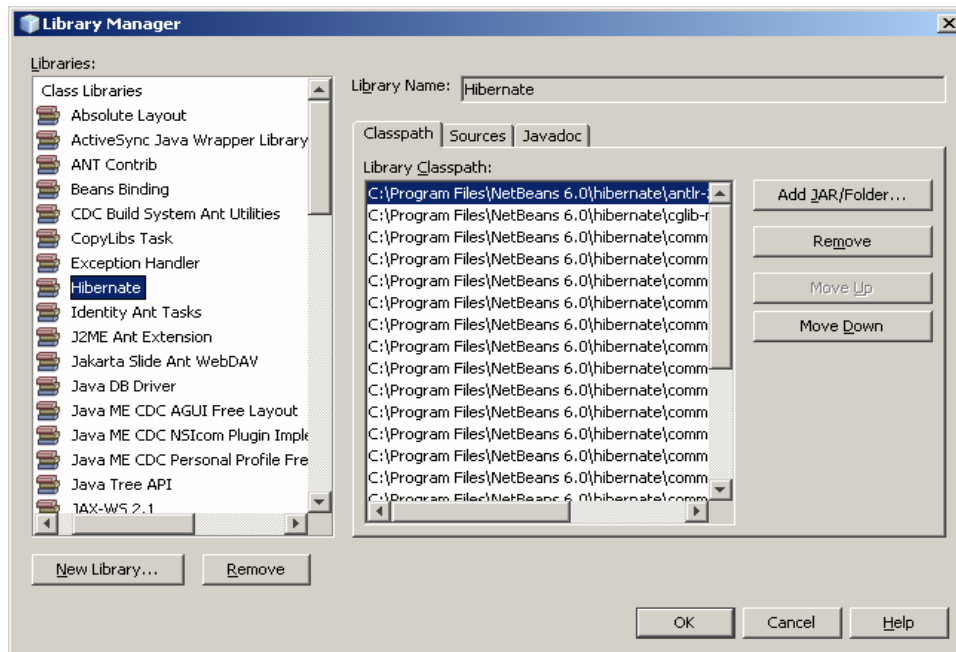
Setelah beberapa artikel tentang PHP sekarang waktunya buat artikel tentang Java... hehe... Pada artikel ini akan membahas tentang dasar membuat proses Create, Read, Update dan Delete menggunakan Hibernate sebagai Framework dan Netbean sebagai IDE-nya, biasanya konfigurasi Hibernate selalu tidak pernah lepas dengan konfigurasi Spring, tapi pada kesempatan ini saya akan menguraikan tentang Hibernate tanpa Spring, dengan tujuan jika ingin membuat aplikasi desktop maka konfigurasi Spring tidak diperlukan.

Sebelum memulai ada baiknya kita mengetahui class-library apa saja yang diperlukan untuk membangun aplikasi jika akan menggunakan Hibernate Annotations sebagai berikut:

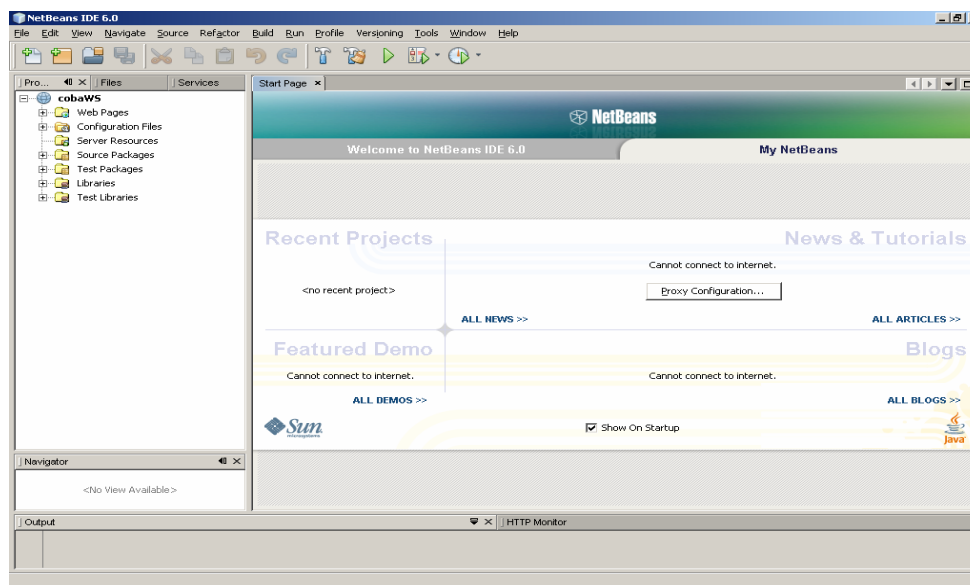


Sekarang bagaimana caranya memasukan library – library tersebut kedalam project properties yang akan kita buat? Cukup kita ikuti langkah – langkahnya sebagai berikut; dari Toolbar pilih Tools diklik, terus akan keluar dropdown menu, kemudian pilih Libraries maka akan keluar window

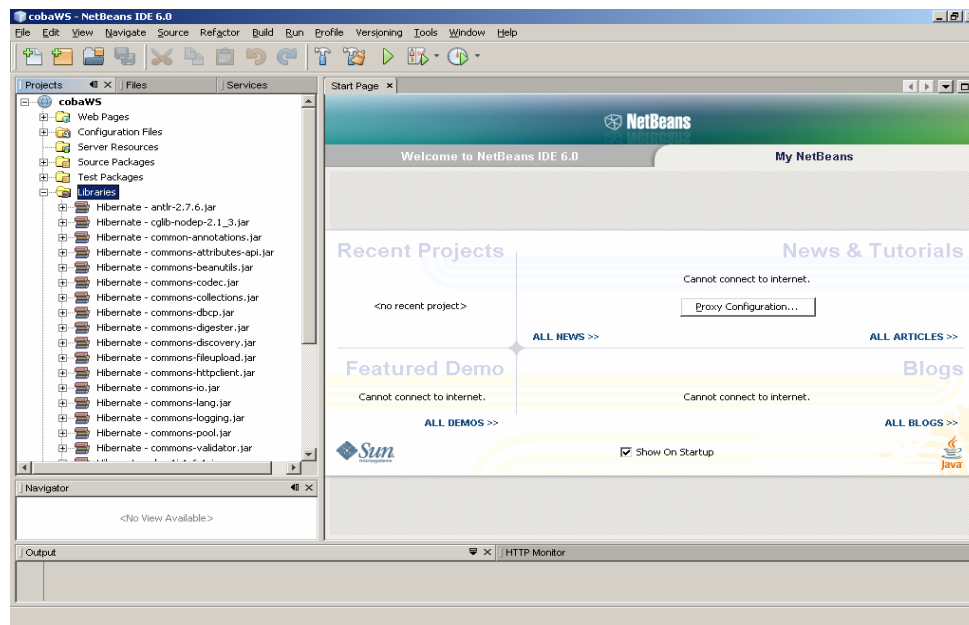
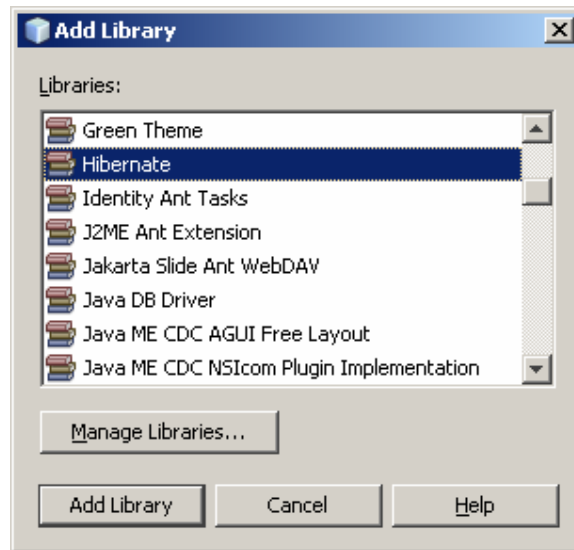
box dengan title "Library Manager", kemudian tekan tombol "New Library...", maka akan keluar lagi window box dengan title "New Library", isikan kata Hibernate pada textbox Library Name, terus pilih "Class Libraries" combobox Library Type, kemudian tekan OK, setelah itu pilih file jar library hibernate seperti yang telah tertera diatas ke folder yang telah kita tentukan sebelumnya dengan cara tekan tombol "Add Jar/Folder..." pada tab "Classpath", setelah file – file terkirim maka akan mempunyai tampilan seperti berikut:



Setelah itu tekan tombol "OK", sekarang pembahasan akan dilanjutkan dengan membuat project baru dalam IDE Netbean. Setelah membuat project baru, baik itu web project atau desktop project maka window netbean akan seperti berikut:



Setelah membuat project, terlebih dahulu memasukan library hibernate dengan cara klik kanan diatas folder library, kemudian klik “Add Library...” dan akan menampilkan window dengan title “Add Library”, diterukan dengan memilih library mana yang diperlukan dipilih kemudian tekan tombol “Add Library”, maka dalam folder Library dalam project akan menampilkan seluruh file jar library yang akan dibutuhkan dalam aplikasi yang akan dibangun.



Setelah seluruh library yang diperlukan disiapkan dalam folder library, sekarang waktunya membahas membuat CRUD menggunakan Hibernate, sebelum mulai coding dengan java class perlu kita ketahui terlebih dahulu basic konfigurasi yang wajib dipenuhi, supaya Hibernate dapat berfungsi dengan baik, konfigurasi ini melibatkan file xml yang akan diletakkan di folder root dari

folder "Source Packages", file ini harus diberi nama "hibernate.cfg.xml" dan isi dari file tersebut akan seperti berikut:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC "-//Hibernate/Hibernate Configuration
DTD 3.0//EN" "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
  <session-factory>
    <!-- Database connection settings -->
    <property
name="connection.driver_class">com.mysql.jdbc.Driver</property>
    <property
name="connection.url">jdbc:mysql://localhost:3306/latih</property>
    <property name="connection.username">root</property>
    <property name="connection.password">admin</property>
    <!-- JDBC connection pool (use the built-in) -->
    <property name="connection.pool_size">1</property>
    <!-- SQL dialect -->
    <property name="dialect">org.hibernate.dialect.MySQLDialect</property>
    <!-- Enable Hibernate's automatic session context management -->
    <property name="current_session_context_class">thread</property>
    <!-- Disable the second-level cache -->
    <property
name="cache.provider_class">org.hibernate.cache.NoCacheProvider</property>
    <!-- Echo all executed SQL to stdout -->
    <property name="show_sql">>true</property>
    <!-- Drop and re-create the database schema on startup -->
    <property name="hbm2ddl.auto">none</property>

    <mapping class="xxx.TabelSiswa"/>
  </session-factory>
</hibernate-configuration>
```

Dari source xml konfigurasi diatas terdapat beberapa properties yang harus ada untuk membangun aplikasi berbasis framework hibernate, berikut daftar beserta keterangannya untuk lebih lengkapnya dapat dilihat pada link http://www.hibernate.org/hib_docs/reference/en/html/session-configuration.html :

- connection.driver.class, property ini digunakan sebagai penentu driver database apa yang akan digunakan untuk tersambung ke database.
- connection.url, property ini digunakan untuk menentukan tujuan url database yang akan terpakai sebagai connection terhadap database.
- connection.username, property user name yang dipakai untuk mengakses database.
- connection.password, property password yang dipakai untuk mengakses database.
- connection.pool.size, property yang digunakan sebagai batasan berapa koneksi maksimum yang boleh digunakan untuk tersambung ke database.
- dialect, property yang menentukan jenis dialect yang akan digunakan, disesuaikan dengan jenis database yang dipakai.
- current_session_context_class, property yang digunakan untuk memperlakukan session context, pilihannya berupa; jta | thread | managed | custom.Class.

- `cache.provider_class`, property untuk menentukan class yang digunakan untuk custom `CacheProvide`.
- `show_sql`, property untuk memperlihatkan perintah sql yang di eksekusi pilihannya bisa true atau false
- `hbm2ddl.auto`, property untuk otomasi validasi atau export schema terhadap database disaat `SessionFactory` dibuat, atau dengan property `create-drop` schema akan langsung di drop ketika `SessionFactory` ditutup. Adapun property diatas menggunakan none berarti tidak melakukan apa – apa terhadap schema database.

Sedangkan untuk baris kode `<mapping class="xxx.TabelSiswa"/>` merupakan perintah untuk mapping suatu class Model Annotation supaya dapat terbaca oleh file `hibernate.cfg.xml`, jumlah mapping ini akan banyak atau sedikit tergantung dari banyaknya class Model yang dibuat untuk menunjang aplikasi yang hendak dibangun tersebut.

Setelah selesai dengan konfigurasi sekarang waktunya membuat class Model yang mewakili sebuah tabel pada database. Sebagai contoh kita akan membuat class “TabelSiswa” yang akan disimpan dalam package “xxx”, berikut kode-nya:

```
package xxx;

import java.io.Serializable;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name = "tbl_siswa")
public class TabelSiswa implements Serializable{
    @Id
    @GeneratedValue
    private int id;

    @Column(name="no_induk", nullable=false,length=10)
    private String nomorInduk;

    @Column(name="nama", nullable=false,length=30)
    private String nama;

    @Column(name="alamat", nullable=false,length=45)
    private String alamat;

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getNomorInduk() {
```

```
        return nomorInduk;
    }

    public void setNomorInduk(String nomorInduk) {
        this.nomorInduk = nomorInduk;
    }

    public String getNama() {
        return nama;
    }

    public void setNama(String nama) {
        this.nama = nama;
    }

    public String getAlamat() {
        return alamat;
    }

    public void setAlamat(String alamat) {
        this.alamat = alamat;
    }
}
```

Kode diatas mewakili sebuah table dalam database MySQL yang bernama “tbl_siswa” dengan nama field yang tertera dalam properties @Column diatas, dari source kode “TabelSiswa” diatas dapat kita lihat beberapa syntax code yang diawali dengan tanda “@”, inilah yang disebut Annotations sekilas akan dijelas arti dari masing – masing perintah diatas sebagai berikut, namun untuk lebih jelasnya dapat dilihat pada link berikut “http://www.hibernate.org/hib_docs/annotations/reference/en/html/entity.html” (this link written in English, so please read it carefully):

- @Entity, berfungsi untuk mendefinisikan sebuah class Model adalah sebuah Entity bean yang dihubungkan dengan POJO persistence.
- @Table, berfungsi untuk menghubungkan suatu class Entity terhadap sebuah table dalam sebuah schema database.
- @Id, berfungsi untuk mendefinisikan salah satu property field dari entity bean yang akan berperan sebagai identifier.
- @GeneratedValue, berfungsi untuk mendefinisikan type generator yang digunakan untuk memperoleh nilai dari identifier @Id.
- @Column, berfungsi sebagai property field mapping terhadap field yang berada di table di database.

Kemudian buatlah file java class yang diberi nama HibernateUtility.java yang digunakan sebagai Session Factory, yang menghubungkan suatu proses transaction yang dibuat secara programmatic dengan system konfigurasi hibernate yang telah ditentukan sebelumnya. Adapun source code dari HibernateUtility akan seperti berikut:

```
package xxx;

import org.hibernate.SessionFactory;
import org.hibernate.cfg.AnnotationConfiguration;

public class HibernateUtil {
    private static final SessionFactory sessionFactory;

    static{
        try{
            sessionFactory = new
AnnotationConfiguration().configure().buildSessionFactory();
        }catch(Throwable th){
            System.err.println("Initial SessionFactory creation failed"+th);
            throw new ExceptionInInitializerError(th);
        }
    }

    public static SessionFactory getSessionFactory(){
        return sessionFactory;
    }
}
```

Setelah selesai membuat HibernateUtility.java sekarang mulai membahas pada proses Create, Read, Update dan Delete, yang akan disimpan dalam package “xxx.client” berikut source kode “CreateData.java” yang berisi contoh perintah untuk menginsert data:

```
package xxx.client;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import xxx.HibernateUtil;
import xxx.TabelSiswa;

public class CreateData {
    public static void main(String[] args) throws Exception {
        SessionFactory sessFact = HibernateUtil.getSessionFactory();
        Session sess = sessFact.getCurrentSession();
        Transaction tr = sess.beginTransaction();
        TabelSiswa stu = new TabelSiswa();
        stu.setNama("Yudhi");
        stu.setNomorInduk("100");
        stu.setAlamat("Jl. Sukajadi No. 10");
        sess.save(stu);
        tr.commit();
        System.out.println("Successfully inserted");
        sessFact.close();
    }
}
```

Membahas apa yang tertulis pada source kode pada baris pertama sampai ketiga dalam method “main” tertulis perintah untuk inialisasi Session Hibernate dan Transaction, kemudian dilanjutkan dengan inialisasi “TabelSiswa” yang menghubungkan langsung dengan tabel “tabel_siswa” di database, dengan mengisikan nilai variable terhadap method – method “set” kemudian dimasukan ke method “save” yang berada di bawah inialisasi Session, hal tersebut sudah merupakan proses “insert into” terhadap tabel di database. Kemudian perintah di tutup dengan

perintah “commit” untuk eksekusi insert data terhadap tabel dan perintah “close” untuk mengakhiri inisialisasi Session. Dilanjutkan dengan “ReadData.java” berikut source kode-nya:

```
package xxx.client;

import java.util.Iterator;
import java.util.List;
import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import xxx.HibernateUtil;
import xxx.TabelSiswa;

public class ReadData {
    public static void main(String[] args) throws Exception {
        SessionFactory sessFact = HibernateUtil.getSessionFactory();
        Session sess = sessFact.getCurrentSession();
        Transaction tr = sess.beginTransaction();
        Query query = sess.createQuery("from TabelSiswa");
        List result = query.list();
        Iterator it = result.iterator();
        System.out.println("id  sname  sroll  scourse");
        while(it.hasNext()){
            TabelSiswa st = (TabelSiswa)it.next();
            System.out.print(st.getId());
            System.out.print("    "+st.getNomorInduk());
            System.out.print("    "+st.getNama());
            System.out.print("    "+st.getAlamat());
            System.out.println();
        }
        sessFact.close();
    }
}
```

Sedikit berbeda dengan source kode sebelumnya, perintah diatas adalah untuk menampilkan data dari tabel atau sama dengan perintah “select” pada query, tetapi jika menggunakan Hibernate cukup dengan perintah “createQuery” dan sebutkan nama instant mapping tabel tersebut sudah cukup untuk mengambil data, kebutuhan pengambilan data dapat di rubah where clause-nya sesuai kebutuhan. Selain itu juga diperlukan class List dan Iterator untuk mengurai class yang terenkapsulasi menjadi object – object yang terkecil. Dilanjutkan dengan “UpdateData.java” berikut source kode-nya:

```
package xxx.client;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import xxx.HibernateUtil;
import xxx.TabelSiswa;

public class UpdateData {
    public static void main(String[] args) throws Exception{
        SessionFactory sessFact = HibernateUtil.getSessionFactory();
        Session sess = sessFact.getCurrentSession();
        Transaction tr = sess.beginTransaction();
```

```
TabelSiswa st = (TabelSiswa)sess.load(TabelSiswa.class,4);
st.setAlamat("Jl. Lodaya No. 125");
tr.commit();
System.out.println("Update Successfully");
sessFact.close();
}
}
```

Untuk source kode UpdateData proses yang terjadi tidak jauh berbeda dengan yang terjadi pada proses CreateData, hal kecil yang membedakan hanya pada baris “TabelSiswa st = (TabelSiswa)sess.load(TabelSiswa.class,4);” yang digunakan untuk mengambil data yang akan diedit ke database dengan entity model “TabelSiswa” kemudian lakukan perubahan, setelah itu lakukan commit. Dan yang terakhir adalah proses delete yang disimpan dalam file “DeleteData.java” yang source codenya sebagai berikut:

```
package xxx.client;

import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import xxx.HibernateUtil;
import xxx.TabelSiswa;

public class DeleteData {
    public static void main(String[] args) throws Exception{
        // TODO Auto-generated method stub

        SessionFactory sessFact = HibernateUtil.getSessionFactory();
        Session sess = sessFact.getCurrentSession();
        Transaction tr = sess.beginTransaction();
        TabelSiswa st = (TabelSiswa)sess.load(TabelSiswa.class,4);
        sess.delete(st);
        System.out.println("Deleted Successfully");
        tr.commit();
        sessFact.close();
    }
}
```

Untuk source kode DeleteData proses yang terjadi tidak jauh berbeda dengan yang terjadi pada proses UpdateData, hal kecil yang membedakan hanya pada baris “sess.delete(st);” yang digunakan untuk mengambil data yang akan dihapus dari database dengan entity model “TabelSiswa”, setelah itu lakukan commit.

Okeh dech selamat mencoba! Mungkin saat ini hanya sekian ilmu yang bisa dibagi dengan pembaca semoga bermanfaat, jika ada kekurangan atau masukan yang dapat meningkatkan kemampuan jangan sungkan untuk memberikan komentar.

Berikut daftar link yang dapat digunakan untuk download jar file yang diperlukan:

- <http://antlr.org/>

- http://www.java2s.com/Code/Jar/Spring-Related/cglib-nodep-2.1_3.jar.htm
- http://sourceforge.net/project/showfiles.php?group_id=56933
- http://sourceforge.net/project/showfiles.php?group_id=40712
- <http://commons.apache.org/downloads/>
- <http://www.dom4j.org/download.html>
- http://sourceforge.net/project/showfiles.php?group_id=93232
- <http://logging.apache.org/log4j/1.2/download.html>