Java Jar Dosyalarının Korunması İçin ProGuard Kütüphanesi Kullanımı

Java'da **jar** ve Android'te **apk** dosyaları oluşturulduğu zaman bu jar dosyaları içerisinde, kaynak kodları olmasa bile, **decompile** işlemine tabi tutulduğunda **.class** dosyalarının içerisinde ne **yazıldığı** açık bir şekilde **görülmektedir.** Bundan dolayı jar veya apk oluşturma işleminde mümkün olduğunca kodların anlaşılmasının zorlaştırılması gerekmektedir. Bunun için Proguard isimli bir kütüphane mevcuttur.

Proguard sitesindeki tanım şu şekildedir:

Proguard is a free Java **class file shrinker, optimizer, obfuscator,** and **preverifier.** It detects and removes unused **classes**, fields, methods, and attributes. It optimizes bytecode and removes unused instructions. It renames the remaining **classes**, fields, and methods **using short** meaningless names. Finally, it preverifies the processed code for Java 6 or higher, or for Java Micro Edition.

Proguard'ı maven uygulamasında kulllanmak için aşağıdaki properties element değerleri ve plugin'i eklenmelidir:

```
<properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <java.bootstrap.classes>
            ${java.home}/lib/rt.jar
        </java.bootstrap.classes>
        <java.cryptographic.extension.classes>
            ${java.home}/lib/jce.jar
        </java.cryptographic.extension.classes>
        <java.secure.socket.extension.classes>
            ${java.home}/lib/jsse.jar
        </java.secure.socket.extension.classes>
</properties>

<plugin>
        <!--groupId>com.pyx4me</groupId-->
        <groupId>com.github.wvengen</groupId>
        <artifactId>proguard-maven-plugin</artifactId>
        <executions>
            <execution>
                <id>proguard</id>
                <phase>package</phase>
                <goals>
                    <goal>proguard</goal>
                </goals>
            </execution>
        </executions>
        <configuration>
            <obfuscate>true</obfuscate>
            <includeDependency>true</includeDependency>
            <injar>${project.artifactId}-${project.version}.jar</injar>
            <outjar>${project.artifactId}-${project.version}.jar</outjar>
            <outputDirectory>${project.build.directory}</outputDirectory>
            <options>
                <option>-allowaccessmodification</option>
                <option>-dontskipnonpubliclibraryclasses</option>
                <option>-dontskipnonpubliclibraryclassmembers</option>
                <option>-keep public class * { public protected *; }</option>
            </options>
            <libs>
                <lib>${java.bootstrap.classes}</lib>
                <lib>${java.cryptographic.extension.classes}</lib>
                <lib>${java.secure.socket.extension.classes}</lib>
            </libs>
```

```
        </configuration>
        <dependencies>
            <dependency>
                <groupId>net.sf.proguard</groupId>
                <artifactId>proguard-base</artifactId>
                <version>4.10</version>
                <scope>runtime</scope>
            </dependency>
        </dependencies>
    </plugin>
```

Not: Bu plugin maven **assembly** veya **shade** plugin ile kullanılmalıdır.

Eğer elinizde çalıştırılabilir bir jar dosyası varsa, bu dosya Proguard ile kullanmak için aşağıdaki adımları uygulayınız:

1. Eğer Java **JDK** 1.7 versiyonu veya üstü kurulu ise Proguard jar 4.10 versionu olmalıdır:

İndirmek için tıklayınız

Eğer **JDK** 1.7 den önceki sürümü varsa 4.3 versiyonu yeterlidir.

2. .pro uzantılı bir ayarlar dosyası oluşturup bu dosyaya elimizde bulunan çalıştırılabilir jar dosyasının ayarlarını girmek gerekmektedir. Örnek bir deneme.pro uzantılı dosya şu şekildedir:

```
1   -injars divide.jar
2   -outjars divide_out.jar
3   -libraryjars <java.home>/lib/rt.jar
4   -printmapping divide.map
5
6   -keep public class mucayufa.java.divideImage.Main {
7       public static void main(java.lang.String[]);
8   }
```

İkinci satırda Proguard'ın oluşturacağı dosya(yeni çalıştırılabilir jar dosyası) belirtilir.

Üçüncü satırda Proguard'ın çalışabilmesi için gerekli olan rt.jar dosyasının yolu belirtilir.(Bu jar dosyasını bilgisayarınızda kurulmuş olan jre'nin adresinde görebilirsiniz.). Bu satırda herhangi bir değişiklik **yapmayın!**

Not: <java.home> java jdk'nın bulunduğu dizindir. Bunun sistem değişkenlerinde tanımlı olması gerekmektedir. Tanımlamak için aşağıdaki yol izlenir:

`Denetim Masası -> Sistem ve Güvenlik -> Sistem -> Gelişmiş Sistem Ayarları -> Çevre Değişkenleri Environment Variables) -> Sistem Değişkenleri -> Yeni -> Değişken Adı: JAVA_HOME, Değişken Değeri: C:\Program Files\Java\jdk{Versiyon numarası} Örnek olarak Değişken Değeri C:\Program Files\Java\jdk1.7.0_45 gibi`

Dördüncü satırdaki ifade ise Proguard'ın oluşturduğu **map** dosyasıdır. Bu dosyada orjinal değişken isimlerinin karşısında bu isimlerin değiştirilmiş hali bulunur.

Altıncı satırda ise değişiklik yapmadan kalacak sınıf belirtiliyor. Proguard bu sınıf dışında kalan sınıflara yeni isimlendirmeler vererek, kodların anlaşılmasını zorlaştırmaktadır.

3. Komut satırından(command prompt) aşağıdaki komut çalıştırılırsa işlem tamamlanmış olur.

```
1   java -jar proguard.jar @deneme.pro
```