Unbekannte Kostbarkeiten des SDK Heute: Base64-Kodierungen

Bernd Müller, Ostfalia

Das Java SDK enthält eine Reihe von Features, die wenig bekannt sind. Wären sie bekannt und würden sie verwendet, könnten Entwickler viel Arbeit und manchmal sogar zusätzliche Frameworks einsparen. Wir wollen in dieser Reihe derartige Features des SDK vorstellen: die unbekannten Kostbarkeiten.

Die Kodierung von Binärdaten in Codepage-unabhängige ASCII-Zeichen wird in der Regel beim Datenaustausch per XML, JSON oder Ähnlichem verwendet. Das Standardformat dafür ist Base64. Seit Java 8 enthält das SDK die Klasse `Base64`, die diese Kodierung zur Verfügung stellt.

Alternative Base64-Kodierungen

Der Autor hat vor vielen Jahren in einem Projekt die letzte vollständig öffentliche Version von iText verwendet, die Version 2.1.7. Diese definierte als Abhängigkeit Bouncy Castle *[1]*, eine Crypto-Bibliothek, die eine Klasse `Base64` zur Kodierung von Binärdaten bereitstellt. Weitere häufig genutzte Implementierungen sind Apache Commons Codec *[2]* oder die nicht öffentlichen SDK-Klassen `sun.misc.BASE64Decoder` und `sun.misc.BASE64Encoder`.

Base64 in Java 8

Mit Java 8 erblickte die Klasse `Base64` im Package `java.util` das Licht der Welt. Laut JavaDoc werden die Base64-Kodierungen der RFCs 4648 *[3]* und 2045 *[4]* durch entsprechende Klassen, nämlich `Base64.Decoder` und `Base64.Encoder`, implementiert. Die RFCs unterscheiden die folgenden Kodierungsarten:

```
public class DeEncodingTest {

    private static final byte[] CAFEBABE = new byte[]{0xC,0xA,0xF,0xE,0xB,0xA,0xB,0xE};
    private static final byte[] ENCODED = Base64.getEncoder().encode(CAFEBABE);

    @Test
    public void encoding() {
  byte[] encodedByJdk = java.util.Base64.getEncoder().encode(CAFEBABE);
  byte[] encodedByBc = org.bouncycastle.util.encoders.Base64.encode(CAFEBABE);
  byte[] encodedByCommons = org.apache.commons.codec.binary.Base64.encodeBase64(CAFEBABE);
  Assertions.assertArrayEquals(encodedByJdk, encodedByBc);
  Assertions.assertArrayEquals(encodedByJdk, encodedByCommons);
    }

    @Test
    public void decoding() {
  byte[] decodedByJdk = java.util.Base64.getDecoder().decode(ENCODED);
  byte[] decodedByBc = org.bouncycastle.util.encoders.Base64.decode(ENCODED);
  byte[] decodedByCommons = org.apache.commons.codec.binary.Base64.decodeBase64(ENCODED);
  Assertions.assertArrayEquals(decodedByJdk, decodedByBc);
  Assertions.assertArrayEquals(decodedByJdk, decodedByCommons);
    }

}
```

Listing 1

- Basic (`getDecoder()`, `getEncoder()`)
- URL und Filename (`getUrlDecoder()`, `getUrlEncoder()`)
- MIME (`getMimeDecoder()`, `getMimeEncoder()`)

Die in den Klammern genannten Klassenmethoden der Klasse `Base64` geben jeweils dedizierte Implementierungen der genannten Kodierungsarten von `Base64.Decoder` und `Base64.Encoder` zurück.

Die Klasse `Base64.Decoder` enthält die Methoden

```
byte[] decode(byte[] src)
int decode(byte[] src, byte[] dst)
byte[] decode(String src)
ByteBuffer decode(ByteBuffer buffer)
InputStream wrap(InputStream is)
```

Die Klasse `Base64.Encoder` enthält die Methoden

```
byte[] encode(byte[] src)
int encode(byte[] src, byte[] dst)
ByteBuffer encode(ByteBuffer buffer)
String encodeToString(byte[] src)
Base64.Encoder withoutPadding()
OutputStream wrap(OutputStream os)
```

Die genannten Methodensignaturen lassen die Verwendungsmöglichkeiten relativ gut erahnen. Dass die grundlegenden Methoden zur Base64-Kodierung des SDK mit denen von Bouncy Castle und Apache Commons Codec übereinstimmen, überprüfen wir mit einfachen Tests, die in der JUnit-Klasse DeEncodingTest in *Listing 1* dargestellt sind. Zur besseren Zuordnung haben wir die verwendeten Klassen mit voll qualifizierten Klassennamen verwendet.

Zusammenfassung

Seit Java 8 erlaubt die Klasse `java.util.Base64` das Kodieren und Dekodieren binärer Daten auf Basis der Base64-Kodierung.

Referenzen

[1] Bouncy Castle. *https://www.bouncycastle.org/*

[2] Apache Commons Code. *https://commons.apache.org/proper/commons-codec/*

[3] The Base16, Base32, and Base64 Data Encodings. *https://www.ietf.org/rfc/rfc4648.txt*

[4] Multipurpose Internet Mail Extensions (MIME). *https://www.ietf.org/rfc/rfc2045.txt*

Bernd Müller
Ostfalia
bernd.mueller@ostfalia.de

Nach seinem Studium der Informatik und der Promotion arbeitete Bernd Müller für die IBM und die HDI Informationssysteme. Er ist Professor, Geschäftsführer, Autor mehrerer Bücher zu den Themen JSF und JPA, sowie Speaker auf nationalen und internationalen Konferenzen. Er engagiert sich im iJUG und speziell in der JUG Ostfalen.