File I/O und XML

SEP WiSe 2012/13
Tobias Fechner und David Jost

Nutzung von File I/O in Java

```
1 import java.io.*;
```

Alles Folgende in Try/Catch aufgrund von IOException

Neues File erstellen:

```
1 File file = new File(DATEINAME ODER PFAD ALS STRING);
```

Datei erstellen:

```
1 boolean file.createNewFile();
```

Datei löschen:

```
1 boolean file.delete();
```

Einen Stream auf einer Datei öffnen (binär):)

```
1 FileInputStream fileinputstream = new FileInputStream(file);
2 FileOutputStream fileoutputstream = new FileOutputStream(file);
```

Den Stream lesen (einen Buffer aufbauen):

```
1 BufferedInputReader bufferedinputreader = new BufferedInputReader(
     fileinputstream);
```

Den Buffer lesen:

```
1 while (bufferedinputreader.available() != 0)
2 {
3   bufferedinputreader.read();
4 }
```

Den Buffer schreiben (binär):

```
1 fileoutputstream.write(contentInBytes);
2 fileoutputstream.flush();
3 fileoutputstream.close();
```

Den Stream auf jeden Fall im `finally` schließen:

```
1 if (fileoutputstream != null)
2 {
3   fileinputstream.close();
4 }
```

Den Stream lesen (Textzeichen):

```
1 BufferedReader bufferedReader = new BufferedReader(new FileReader(file));
2 while((String line = bufferedReader.readLine()) != null)
3 {
4   ...();
5 }
```

Einen Stream schreiben (Textzeichen):

```
1 BufferedWriter bufferedWriter = new BufferedWriter(new FileWriter(file));
2 bufferedWriter.write(STRING);
3 bufferedWriter.close();
```

Aufbau eines XML-Dokumentes

```
<?xml version="1.0" encoding="utf-8" ?>
<!DOCTYPE root SYSTEM "root.dtd">
<!-- Kommentar -->
<root>
  <subelement id="sub_1">
    <sub2 name="SUB2_name" >TEXT 1</sub2>
    <sub3 title="Text mit Whitespaces" />
  </subelement>
  <subelement id="sub_2">
    <sub2 name="SUB2_name" >TEXT 2</sub2>
    <sub3 title="stand-alone-tag" />
    <sub3 title="ein weiteres" />
  </subelement>
</root>
```

Dazugehöriges Schema (DTD, root.dtd):

```
<?xml version="1.0" encoding="utf-8" ?>
<!ELEMENT root (subelement)>
<!ELEMENT subelement (sub2,sub3+)>
<!ELEMENT sub2 (#PCDATA)>
<!ELEMENT sub3 EMPTY>
<!ATTLIST subelement id ID #REQUIRED>
<!ATTLIST sub2 name NMTOKEN #IMPLIED>
<!ATTLIST sub3 title NMTOKENS "none">
```

XML in Java mit JAXB

```
import javax.xml.bind.JAXBContext; //um JAXB zu nutzen
import javax.xml.bind.Marhaller/Unmarshaller; //zum Lesen/Schreiben
import javax.xml.bind.annotations.*; //fuer die Klassen
```

XML-Datei öffnen und einlesen:

```
try {
  File file = new File(XMLPATH);
  JAXBContext jaxbContext = JAXBContext.newInstance(Root.class);
  Unmarshaller jaxbUnmarshaller = jaxbContext.createUnmarshaller();
  _root = (Root) jaxbUnmarshaller.unmarshal(file);
} catch (Exception e) { ... }
```

Daten in der XML-Datei speichern:

```
try {
  File file = new File(XMLPATH);
  JAXBContext jaxbContext = JAXBContext.newInstance(Root.class);
  Marshaller jaxbMarshaller = jaxbContext.createMarshaller();
  jaxbMarshaller.setProperty(Marshaller.JAXB_FRAGMENT, true);
  jaxbMarshaller.setProperty("com.sun.xml.internal.bind.xmlHeaders", "<?
     xml version=\"1.0\" encoding=\"utf-8\" ?>\n<!DOCTYPE root SYSTEM \"
     root.dtd\">");
  jaxbMarshaller.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT, true);
  jaxbMarshaller.marshal(_root, file);
} catch (Exception e) { ... }
```

Annotations in den entsprechenden Klassen (an Getter und Setter):

`@XmlRootElement([name="..."])`	`@XmlElement([name="..."])`
`@XmlAttribute([name="..."])`	`@XmlValue`
`@XmlElementWrapper([name="..."])`	`@XmlTransient`

Hilfreich dafür: `http://openbook.galileodesign.de/javainsel8/javainsel_15_008.htm`