Formale Definition von `pathname` in EBNF

Im folgenden definieren wir formal den Begriff `pathname` mit Hilfe der EBNF.

Die verwendete EBNF-Notation folgt dabei dem ISO-Standard ISO_IEC_14977_1996E.
Entgegen den vielfach kursierenden Varianten der EBNF, müssen die syntaktischen Terme gemäß Standard durch ein Komma separiert werden.
Um im Text leichter auf die einzelnen Produktionen verweisen zu können, sind die Produktionen in Kommentaren nummeriert (`P1-P9`).

```
(* P1 *) pathname = path , filename ;
(* P2 *) path = [drive] , { name , delimiter } ;
(* P3 *) drive = letter , ':' ;
(* P4 *) filename = name ;
(* P5 *) delimiter = '/' | '\' ;
(* P6 *) name = { letter | digit | special | '.' } ;
(* P7 *) special = ' ' | '-' | '_' ;
(* P8 *) digit  = ? alle Zeichen c für die Character.isDigit(c) == true ? ;
(* P9 *) letter = ? alle Zeichen c für die Character.isLetter(c) == true ? ;
```

Erläuterungen zu den einzelnen Produktionen:

zu P2 :

Das Nicht-Terminal `name` (P6) kann auch zu einer leeren Zeichenkette expandiert werden. Daher kann man beliebig lange (auch leere) Sequenzen der Terminale `'/'` und `'\'` in einem `path` (P2) erzeugen.

Beispiele:

```
d:/part1//part2/\\part3///   (in Windows für 'cd' erlaubt)
d:
d:part1\part2\               (in Windows für 'cd' aus d:\ erlaubt)
```

Insbesondere kann `path` auch zur leeren Zeichenkette expandiert werden.

Zu P7:

Das erste Terminal der Produktion ist ein Leerzeichen (ein Space)

Zu P8 und P9:

Wir nutzen hier die Möglichkeit der EBNF, über ? … ? externe Definitionen für Nicht-Terminale einzubinden. Hier wird auf die Java-Methoden

`Character.isDigit(c)` und `Character.isLetter(c)`

Bezug genommen.

Bemerkung:
Herkömmliche Betriebssysteme wie Windows oder Linux erlauben zum Teil sogar eine noch allgemeinere Syntax für `pathname`. Insbesondere sind noch mehr Sonderzeichen erlaubt (P7).

Beispiel (Linux): `' Das ist //#möglich / aber .... /nicht%sinvoll   . mp3 '`