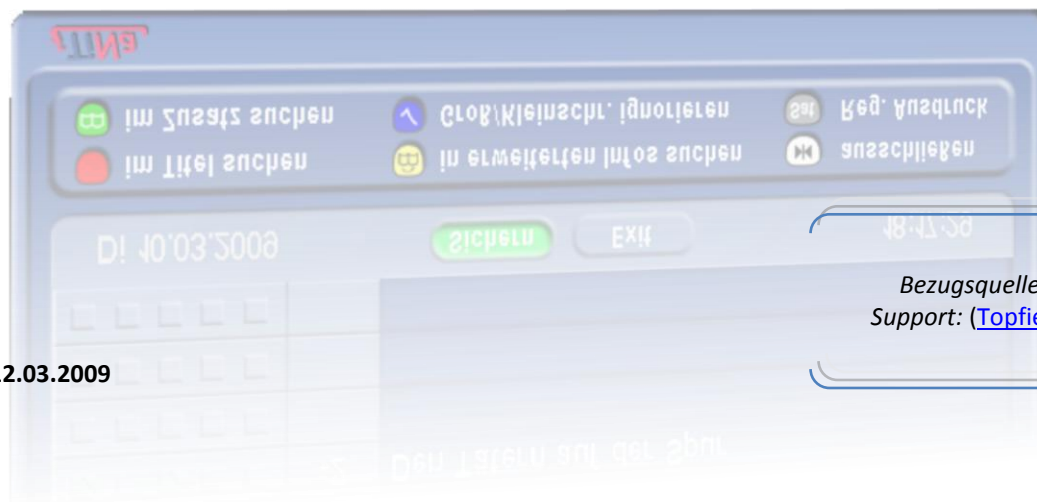
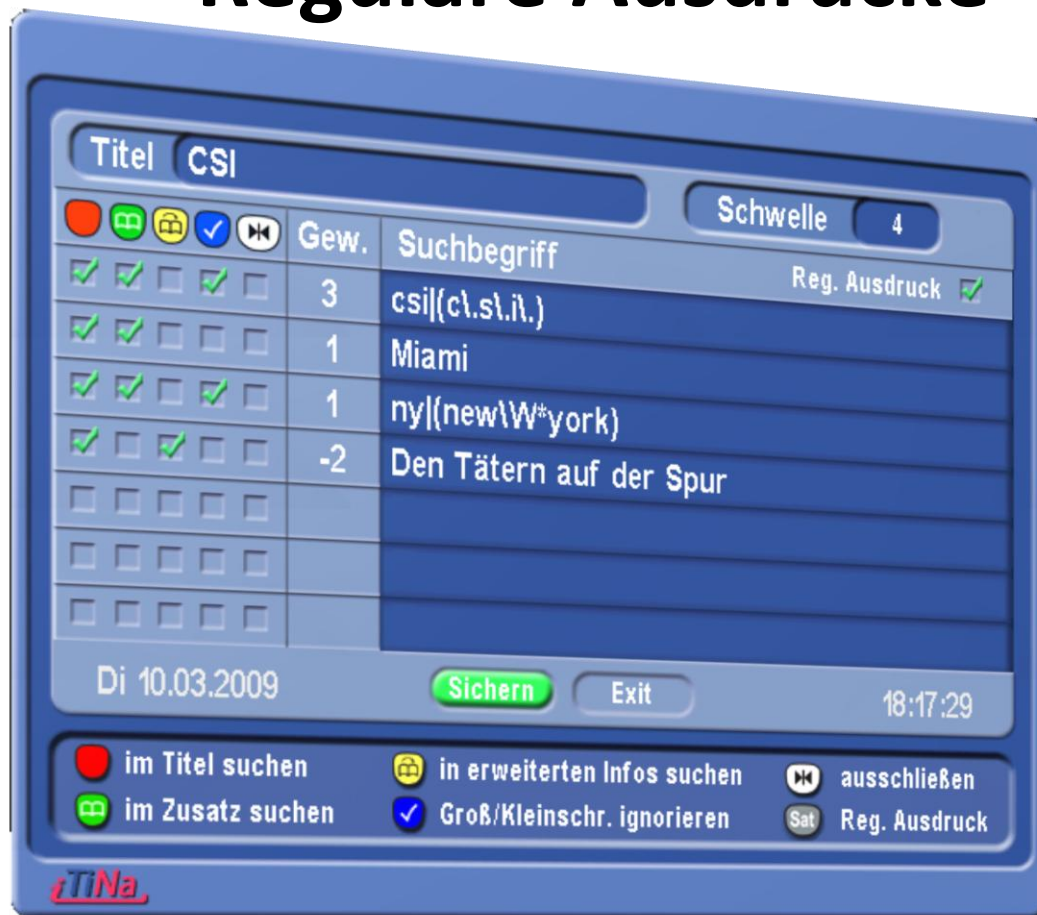


intelligent **T**imer & **N**avigation

# iTiNa 1.02

## Reguläre Ausdrücke



Bezugsquelle: ([iTiNa](#))  
Support: ([Topfield-Forum](#))

12.03.2009

# Inhaltsverzeichnis

1.	Einleitung.....	3
2.	Begriffe und Sonderzeichen bei Regulären Ausdrücken .....	4
2.1	Einzelzeichen (CHAR) .....	4
2.2	Zeichenketten (STRG) .....	4
2.3	Zeichenklasse .....	5
2.4	Besondere Zeichenklassen.....	5
2.5	Weitere Zeichenklassen .....	6
2.6	Klammern.....	6
2.7	Beliebiges Zeichen .....	6
2.8	Maskierung von Sonderzeichen.....	6
2.9	Wiederholungen .....	7
2.10	Alternativen .....	8
2.11	Wortgrenzen .....	8
2.12	Keine Wortgrenzen .....	9
2.13	Zeilenanfang.....	9
2.14	Zeilenende .....	9
3.	Ein paar Beispiele zum Üben .....	10
4.	Reguläre Ausdrücke an einem Beispiel aufgeschlüsselt.....	12
5.	ASCII-Tabelle.....	14

# 1. Einleitung

---

iTiNa bietet die Möglichkeit, den EPG nach Begriffen zu durchsuchen. Im einfachsten Fall wird dabei mit Hilfe der manuellen Suche nach einem einzelnen Begriff gesucht. Dieser könnte z.B. "Die vier Elemente" lauten. Das Ergebnis der Suche wird anschließend in einer Trefferliste angezeigt.

Darüber hinaus bietet iTiNa die Möglichkeit, komplexe Suchaufträge, so genannte Tinas anzulegen. Hierbei können bis zu sieben einzelne Suchbegriffe in einem Tina verwendet werden. Die Suchbegriffe selbst können dabei "Reguläre Ausdrücke" enthalten. Und darum geht es in diesem Handbuch.

Was sind nun "Reguläre Ausdrücke"? Schauen wir uns die Definition aus Wikipedia an:

*„In der Informatik ist ein **Regulärer Ausdruck** (Abk. **RegExp** oder **Regex**, engl. *regular expression*) eine Zeichenkette, die der Beschreibung von Mengen beziehungsweise Untermengen von Zeichenketten mit Hilfe bestimmter syntaktischer Regeln dient.“*

Die Beschreibung ist zwar furchtbar richtig, aber im ersten Moment kann man damit nicht viel anfangen. Bei regulären Ausdrücken geht es darum, dass bestimmte Begriffe (Zeichenketten) mit Hilfe von Sonderzeichen (syntaktischen Regeln) beschrieben werden. Und alles was auf diese Beschreibung zutrifft, ist das Ergebnis (die Menge).

Im Grunde ist das erste Beispiel "Die vier Elemente" bereits ein regulärer Ausdruck. Denn er beschreibt eine ganz bestimmte Zeichenfolge, nach der gesucht wird. Jedes Leerzeichen, jeder Buchstabe sowie die Groß-/Kleinschreibung wird dabei berücksichtigt. Möchte man jetzt nicht exakt danach suchen, sondern bestimmte Variationen zulassen, hat man bei iTiNa die Möglichkeit dazu.

## 2. Begriffe und Sonderzeichen bei Regulären Ausdrücken

---

### 2.1 Einzelzeichen (CHAR)

Ein Einzelzeichen ist z.B. ein Buchstabe, eine Ziffer oder ein beliebiges Zeichen. Es kann auch als Oktalzahl angegeben werden → siehe dazu Kapitel 5.

### 2.2 Zeichenketten (STRG)

Eine Zeichenkette ist z.B. "blabla" oder "Die vier Elemente". Die Leerzeichen im zweiten Beispiel trennen dieses nicht in drei einzelne Zeichenketten auf! Alles zwischen den beiden Anführungszeichen ist eine Zeichenkette.

Die Anführungszeichen werden in iTiNa nicht eingegeben, sie dienen hier nur zur Abgrenzung der Suchbegriffe. In der erweiterten Eingabe steht für jeden Suchbegriff eine eigene Zeile zur Verfügung.

Getrennt werden Zeichenketten nur durch Sonderzeichen wie `.` `()` `?` `|` `[]` `{}` `*` `\w` `\:` usw. (alles was mehr als 1 Zeichen beschreibt). D.h. sämtliche danach angegebenen Parameter wie z.B. `?*{2}` beziehen sich auf den kompletten Teil davor.

**Wichtig:** Das ist ein Unterschied zur üblichen Syntax von regulären Ausdrücken. Dort würde sich z.B. ein `"?"` nur auf das Einzelzeichen direkt davor beziehen. Bei iTiNa bezieht es sich auf die Zeichenkette davor!

Beispiel: Möchte man einen Suchbegriff "Wetten, dass" erzeugen, in dem das Komma optional ist, wäre demzufolge "Wetten[,]? dass" nötig. Durch das `[ ]` wird das Komma als Zeichenklasse betrachtet und das `"?"` bezieht sich nur auf das durch die Zeichenklasse angegebene Zeichen. "Wetten,? dass" würde dagegen "Wetten," als optional betrachten.

Werden in dem Begriff Sonderzeichen maskiert (s.u.), trennt das die Zeichenkette nicht auf. Der Suchbegriff "Dr. House" oder "Hallo? Jemand da?" ist eine Zeichenkette.

## 2.3 Zeichenklasse

Zeichenklassen werden in "[ ]" geschrieben und definieren immer 1 Zeichen. Innerhalb der Klammern kann man alle beliebigen Zeichen angeben. So darf z.B. bei "[ae]" ein "a" oder ein "e" vorkommen. Im Suchbegriff würde das dann z.B. so aussehen: "Sepp M[ae][iy]er".

"Sepp M" ist dabei eine Zeichenkette, danach folgen die Zeichenklassen "[ae]" (ein "a" oder ein "e") und "[iy]" (ein "i" oder ein "y") und dann die Zeichenkette "er". Damit kann man "Sepp Meier", "Sepp Mayer", "Sepp Meyer" oder "Sepp Maier" finden, ohne alle Schreibweisen einzeln eingeben zu müssen.

Zur Erleichterung können fortlaufende Folgen, beispielsweise mit "[a-d]" (Buchstaben "a" bis "d") angegeben werden. Das Ganze auch in Kombination, z.B. "[a-dhiABE-H15-8]". Das entspricht einem der Zeichen "a", "b", "c", "d", "h", "i", "A", "B", "E", "F", "G", "H", "1", "5", "6", "7" oder "8".

Möchte man eine Zeichenklasse ins Gegenteil umkehren, muss als erstes Zeichen in der Zeichenklasse ein "^" geschrieben werden. Ein "[^acf]" steht daher für alle Zeichen, außer "a", "c" und "f".

## 2.4 Besondere Zeichenklassen

Es gibt eine Reihe von vordefinierten Zeichenklassen:

- [ :upper: ]** Großbuchstaben, Umlaute und Sonderbuchstaben bestimmter Sprachen (z.B. Á, Û, Ï, É usw.).
- [ :lower: ]** Kleinbuchstaben, Umlaute und Sonderbuchstaben bestimmter Sprachen (z.B. é, å, à, ç, ß usw.).
- [ :alpha: ]** Kleine und große Buchstaben bzw. Umlaute und Sonderbuchstaben: **[ :lower: ]** und **[ :upper: ]**.
- [ :digit: ]** Ziffern: 0, 1, 2, ... bis 9.
- [ :alnum: ]** Alphanumerische Zeichen: **[ :alpha: ]** und **[ :digit: ]**.
- [ :punct: ]** Zeichen wie: ! " # \$ % & ' ( ) \* + , - . / : ; < = > ? @ [ \ ] ^ \_ ` { | } ~ .
- [ :graph: ]** Grafische Zeichen: **[ :alnum: ]** und **[ :punct: ]**.
- [ :print: ]** Druckbare Zeichen: **[ :graph: ]** und Leerzeichen.
- [ :blank: ]** [Leerzeichen](#) und [Tabulator](#).
- [ :space: ]** [Whitespace](#): tab, newline, vertical tab, form feed, carriage return and space.
- [ :cntrl: ]** [Steuerzeichen](#). Im [ASCII](#)-Code sind das die Zeichen 00 bis 1F, und 7F (DEL).
- [ :xdigit: ]** Hexadezimale Ziffern: 0 bis 9, A bis F, a bis f.

## 2.5 Weitere Zeichenklassen

Folgende Zeichenklassen definieren eine bestimmte Gruppe von Zeichen:

- w** steht für alle Buchstaben und Umlaute, groß und klein
- W** steht für das Gegenteil, also alles außer Buchstaben und Umlauten, groß und klein
- s** steht für Leerzeichen, Tabulator, Zeilenwechsel etc. (alle nicht sichtbaren Zeichen)
- S** steht für das Gegenteil, also alles außer Leerzeichen, Tabulator, Zeilenwechsel etc.
- d** steht für Ziffern
- D** steht für das Gegenteil, also alles außer Ziffern
- :** in einer Zeichenklasse, also "[:]", steht für alle Nichtwortzeichen, also alles außer "a-z", "A-Z", "0-9", "\_" (Unterstrich) und "-" (Bindestrich)

## 2.6 Klammern

Teile eines RegExe können mit runden Klammern "( )" zusammengefasst werden, zum Beispiel, um Wiederholungen auf Teilausdrücke anwenden zu können. Sucht man nach "**Du sagst immer nur (bla(,)?\s+){2,}**", so können so viele "bla" mit Komma und/oder Leerraum dazwischen kommen, wie nötig – mindestens aber 2. Weitere Beispiele gibt es in Kapitel 3.

## 2.7 Beliebiges Zeichen

Ein beliebiges Zeichen (ANY) wird durch einen Punkt "." definiert. Der Ausdruck "**F.nd**" würde z.B. "**Fund**", "**Dosenpfand**" oder "**Pfändung**" finden. Aber beispielsweise nicht "**Hand**", "**Sand**", "**Feind**" oder "**Elefanten**".

## 2.8 Maskierung von Sonderzeichen

Mit dem Backslash "\" vor einem **Sonderzeichen** kann dessen Sonderbedeutungen abgeschaltet werden. Folgende Zeichen müssen maskiert werden, wenn sie nur für sich selbst stehen sollen:

. \ [ ] { } ( ) \* + ? | ^ \$

Will man z.B. exakt "Dr. Who?" finden, muss man "**Dr\ Who\?**" als Suchbegriff eingeben, da "." und "?" Sonderzeichen sind.

**Hinweis 1:** Sonderzeichen müssen in der erweiterten Eingabe nur dann selbst maskiert werden, wenn das Häkchen für reguläre Ausdrücke gesetzt ist. Ohne das Häkchen und in der einfachen Eingabe maskiert iTiNa alle Sonderzeichen automatisch.

**Hinweis 2:** Ein Anführungszeichen muss in iTiNa nicht maskiert werden, da es als ganz normales Zeichen gilt. Wird die Datei mit den Tinas (iTina\_Search.add/act) jedoch am Computer bearbeitet, ist das anders. Dort wird das Anführungszeichen als Trennzeichen verwendet und muss daher mit dem Backslash maskiert werden. Z.B.:  
Tina=TV,N:"Witzig",S:IGN TIT,0,"mehr \ "Fun\\"",0,P:1-5,All,R,240,5,15

## 2.9 Wiederholungen

Wiederholungsdrücke beziehen sich immer auf das, was unmittelbar davor steht (Zeichenkette, Zeichenklasse usw.). In geschweiften Klammern "{}" wird angegeben, wie oft etwas vorkommen muss "{min,max}":

"{8}" Es muss genau 8-mal vorkommen

"{2,9}" Es muss mindestens 2-mal und darf maximal 9-mal vorkommen

"{,4}" Es kann gar nicht vorkommen, darf aber maximal 4-mal vorkommen

"{3,}" Es muss mindestens 3-mal, darf aber beliebig oft vorkommen

"bla{2}" Bedeutet demnach "blabla"

"\w{8,}" Steht für mindestens 8 Buchstaben

"\*" steht für 0-mal bis beliebig oft, entspricht also "{0,}".

Zum Beispiel steht "." für keine oder beliebig viele Zeichen, ein "\w\*" für keinen oder beliebig viele Buchstaben.

Der Ausdruck "Ja\*" steht für beliebig viele oder keine Zeichenfolge "Ja" (also "Ja", "JaJa", "JaJaJa", ...).

"+" steht für 1-mal bis beliebig oft, entspricht also "{1,}".

Zum Beispiel steht "[alpha:]+" für mindestens einen oder beliebig viele Buchstaben.

Ein "\d+" bedeutet, es muss mindestens eine Ziffer vorkommen.

"?" steht für 0-mal bis 1-mal, entspricht also "{,1}".

Zum Beispiel steht "\s?" für ein Leerzeichen, Tabulator usw., dieses muss aber nicht vorkommen.

Bei "wä(h)?re" darf in dem Wort ein "h" vorkommen, muss aber nicht (also "wäre" oder "währe").

"\"=" ist anders als alle anderen Wiederholungen! Es bezieht sich nicht auf die Zeichenkette, sondern nur auf das einzelne Zeichen davor. Es steht für das letzte Zeichen noch einmal.

Zum Beispiel steht "\w\"=" für zwei gleiche Buchstaben hintereinander (z.B. "nn", "TT" oder "ää"). Auch wenn Zeichenketten geklammert sind (s.o.), bezieht sich "\"=" nur auf das Zeichen davor!

Ein "(ja)\=" findet nur "jaa".

## 2.10 Alternativen

Für Alternativen verwendet man das Zeichen "|", es hat die Bedeutung einer ODER-Verknüpfung. Der Ausdruck "**Fisch|Fleisch**" findet alles wo "Fisch" oder "Fleisch" drin vorkommt. Diese Methode sollte nicht an Stelle von Zeichenklassen angewendet werden, da sie zeitfressender ist. Also lieber "[ae]" für "a" oder "e", anstelle von "a|e" verwenden.

Soll innerhalb eines Suchbegriffs ein Teil alternativ gemacht werden, so muss dieser in runden Klammern geschrieben werden. Ansonsten würde alles vor und hinter dem "|" ODER-Verknüpft:

Falsch: "**Das ist|war ein toller Tag**" → Findet "Das ist" oder "war ein toller Tag"

Richtig: "**Das (ist|war) ein toller Tag**" → Findet "Das ist ein toller Tag" oder "Das war ein toller Tag"

## 2.11 Wortgrenzen

Eine Wortgrenze wird definiert durch "\b" oder "\b". Hiermit kann man im Suchbegriff Anfang und Ende eines Wortes festlegen. Der gesuchte Begriff wird nur dann gefunden, wenn er an einer Wortgrenze steht. Das muss kein Leerzeichen sein, es kann auch ein Punkt, ein Komma oder ein anderes Nichtwortzeichen sein.

Zu den sog. Wortzeichen gehören die Zeichen "a-z", "A-Z", "0-9", "\_" (Unterstrich) und "-" (Bindestrich). Dementsprechend bezeichnet man alle anderen Zeichen als Nichtwortzeichen.

Beispiele (Groß-/Kleinschreibung wird ignoriert):

"**nuss**" findet z.B. "Erd**nuss**nase"

"\b**nuss**" findet z.B. "die **Nuss**nase hat ein" oder "die 3.**Nuss**sorte ist", aber nicht "die Erd**nuss** von"

"**nuss**:" findet z.B. "eine Erd**nuss** im" oder "die Wal**nuss**, die im Baum", aber nicht "die **Nussernte** war"

"\b**nuss**:" findet z.B. "die **Nuss** hat", aber nicht "Erd**nuss**nase", "die **Nussernte**" oder "Erd**nuss** im"

## 2.12 Keine Wortgrenzen

Das Gegenteil von "\b" kann mit "\B" erreicht werden. Der gesuchte Begriff wird nur dann gefunden, wenn er sich nicht an einer Wortgrenze befindet. Auch hier wieder ein paar Beispiele:

"\Bnuss" findet z.B. "Erdnuss" oder "Walnuss", aber nicht "die Nussnase" oder "die 3.Nussorte"

"nuss\B" findet z.B. "die Nussnase" oder "das Walnuss-Eis", nicht aber "die Erdnuss hat ein Loch"

## 2.13 Zeilenanfang

"^" am Anfang eines Ausdrucks führt dazu, dass der Begriff nur zu einem Treffer führt, wenn er am Anfang einer Zeile gefunden wird. Ein "^Maus" findet "Maus" nur, wenn es am Anfang einer Zeile steht. So würde z.B. "Maus und Katze" gefunden, "Die Sendung mit der Maus" aber nicht.

## 2.14 Zeilenende

"\$" am Ende eines Ausdrucks führt dazu, dass der Begriff nur zu einem Treffer führt, wenn er am Ende einer Zeile gefunden wird. Ein "Haus\$" findet "Haus" nur, wenn es am Ende einer Zeile steht. So würde z.B. "Mein neues Haus" gefunden, "Ein Haus mit Fenstern" aber nicht.

## 3. Ein paar Beispiele zum Üben

---

"**CSI.\*(Miami|NY)**" findet "CSI", gefolgt von beliebig vielen (oder keinem) Zeichen, gefolgt von "Miami" oder "NY". Z.B. "CSI: Miami" oder "CSI ermittelt in NY" usw.

"**Die \d{2} ?Elemente**" findet "Die ", gefolgt von 0 bis 2 Ziffern, danach ein oder kein Leerzeichen und dahinter "Elemente". Also zum Beispiel "Die Elemente" oder "Die 4 Elemente" oder "Die 10 Elemente" usw.

"**Chili(s)? und Briegel(s)?**" findet "Chili und Briegel" genauso wie "Chilis und Briegels". Der Ausdruck "**(s)?**" macht das "s" alternativ. Es darf vorkommen, muss aber nicht.

"**St(ö|oe)rteb(e|ä|ae)c?ker**" findet "Störtebeker", "Störtebecker", "Störtebäker", "Stoertebecker", "Störtebäcker" usw.

"**Alles[:~]+au(ss|0337)er Sex**" findet "Alles: außer Sex" genauso wie "Alles - außer Sex" oder "Alles, ausser Sex" oder "Alles ausser Sex" etc.

"**Alles**" steht für die Zeichenkette "Alles".

"**[:~]+**" beschreibt alle Nichtwortzeichen und den Bindestrich. Davon muss mindestens ein Zeichen vorkommen, es dürfen aber beliebig viele sein.

"**au(ss|0337)er Sex**" steht für "ausser Sex" oder "außer Sex".

"**\bLesch\b**" findet "Lesch" als einzelnes Wort, also z.B. nicht "Salesch" oder "Leschke".

"**[:;]-[()PpD]**" findet alles was mit ":" oder ";" beginnt, dann ein "-" und dann "(", ")", "P", "p" oder "D". Der Eintrag findet also alle Smilies :-).

"**\([^\(\)]\*\)**" findet alles, was in Klammern steht, bei verschachtelten Klammern nur die innere Klammer.

"**\(**" Es muss eine Klammer gefunden werden

"**[^\(\)]\***" Es darf keine Klammer gefunden werden. Damit wird alles gesammelt, bis eine Klammer gefunden wird.

"**\)**" Es muss eine sich schließende Klammer gefunden werden.

"**\:[^WrL]and**" findet alle Zeichenketten die mit einer Wortgrenze anfangen, deren erstes Zeichen kein "W", "r" oder "L" ist und danach ein "and" kommt. Das könnte z.B. "die 3.**Hand** im", "eine **Sand**uhr hat" oder "wieder **Band**salat" sein.

Folgendes würde dagegen nicht gefunden:

"neuer Landkreis mit" Der Buchstabe "L" ist vor dem "and" nicht erlaubt.

"der Dosenpfand ist" Der Teil mit "fand" ist zwar erlaubt, aber davor muss eine Wortgrenze sein.

"." findet alle Sendungen, da nur ein einziges, beliebiges Zeichen vorkommen muss.

Auf den ersten Blick erscheint so ein Suchbegriff unsinnig, da er alles finden würde. Kommt aber zu einer bestimmten Uhrzeit eine Sendung, die immer einen anderen Titel hat, macht der Suchbegriff wieder Sinn. So könnte z.B. Montags um 20:15 Uhr ein Krimi aufgenommen werden, der jede Woche einen anderen Titel hat.

Im Tina würde man den Sender, den Wochentag und die Uhrzeit (z.B. 20:05 – 20:25) einschränken. Das Ergebnis ist ein einziger Treffer: die Sendung die um diese Uhrzeit kommt. Das hätte gegenüber einem Wiederholungstimer den Vorteil, dass der vom Tina angelegte Timer (und damit die Aufnahme) den Titel der Sendung und die richtige Länge hat.

"(Zug [z]?um\s){2}" findet z.B. "Zug um Zug zum Königsdiplom"

Zitat von Grubix: „Regexe sind stur: Sie suchen genau das, was man ihnen aufträgt.“

...nachdem er feststellen musste, dass "(Zug [z]?um){2}" nicht das tat, was er wollte...  
warum nur? :-)

## 4. Reguläre Ausdrücke an einem Beispiel aufgeschlüsselt

---

Kommen wir nun zu einem Beispiel, wie aus einem sehr strengen Suchbegriff ein deutlich toleranterer gemacht werden kann. So etwas ist insbesondere dann notwendig, wenn die Schreibweise einer Sendung variieren kann.

Ein schönes Beispiel dafür ist die Sendung "Wetten, dass ..?" im ZDF. Hier sieht man in den Zeitschriften und anderen Medien immer wieder andere Schreibweisen. Man könnte zwar einfach nur nach "Wetten" im ZDF suchen, aber das würde sicherlich viele ungewollte Treffer bringen.

Fangen wir an: in der normalen Eingabe für Suchbegriffe würde iTiNa die Zeichen, die bei regulären Ausdrücken eine Sonderbedeutung haben, automatisch maskieren. In der erweiterten Eingabe müssen wir dies selbst erledigen, wenn wir reguläre Ausdrücke verwenden.

Wie wir im vorherigen Kapitel gesehen haben, haben der Punkt und das Fragezeichen bei regulären Ausdrücken eine Sonderbedeutung. Diese müssen mit dem umgekehrten Schrägstrich "\ (Backslash) ausgeschaltet werden.

Aus "Wetten, dass ..?" wird somit **"Wetten, dass \.\.?"**

Es wird aber immer noch exakt dasselbe gesucht.

Wenn im EPG jetzt beispielsweise das Komma weggelassen würde, ergäbe der Suchbegriff keinen Treffer mehr. Sorgen wir also dafür, dass das Komma vorkommen darf, aber nicht muss:

**"Wetten(,)? dass \.\.?"**

Damit nur das Komma alternativ ist, wurde es in Klammern gesetzt und mit dem Fragezeichen versehen. Ohne die Klammern würde mit dem Fragezeichen auch das Wort "Wetten" alternativ gemacht, und das wollen wir ja nicht.

Was passiert aber, wenn im EPG für das "dass" die alte Schreibweise mit einem "ß" verwendet wird? Kein Problem, machen wir beide Schreibweisen möglich:

**"Wetten(,)? da(ss|ß) \.\.?"**

Mit "(ss|ß)" muss entweder ein "ss" oder "ß" vorkommen. Nun haben wir aber auf der Tastatur von iTiNa kein "ß". Keine Sorgen, jedes Zeichen kann auch durch seinen ASCII-Code dargestellt werden. Der ASCII-Code für "ß" ist \0337:

**"Wetten(,)? da(ss|\0337) \.\.?"**

Sieht doch schon ganz gut aus. Kümmern wir uns jetzt um den hinteren Teil. Hier sieht man leider sehr viele Varianten. Mal mit zwei, mal mit drei Punkten. Mal mit, mal ohne ohne Leerzeichen. Was tun? Sie werden es sich schon gedacht haben, auch hierfür bieten reguläre Ausdrücke eine Lösung.

Fangen wir mit den Punkten an. Bisher müssen es genau zwei sein. Sorgen wir dafür, dass es zwei oder drei sein können:

```
"Wetten(,)? da(ss|0337) \.{2,3}?"
```

Der Teil "\." wurde durch "\.{2,3}" ersetzt. Da es mehrere Möglichkeiten gibt eine Wiederholung zu kennzeichnen, könnten Sie z.B. mit dem Plus "+" etwas Ähnliches erreichen. Mit "{2,3}" müssen es aber genau zwei bis drei Punkte sein.

Rücken wir nun den Leerzeichen auf den Leib. Sorgen wir zuerst dafür, dass das Leerzeichen hinter dem "da" nicht vorkommen muss (aber darf). Die Sonderbedeutung des Fragezeichens haben wir dafür ja schon kennen gelernt:

```
"Wetten(,)? da(ss|0337) ?\.{2,3}?"
```

Haben Sie es gefunden? Zwischen dem Leerzeichen und dem Backslash ist das Fragezeichen hinzugekommen.

Da man auch häufiger sieht, dass bei "da.. ?" ein Leerzeichen zwischen den Punkten und dem Fragezeichen ist, fügen wir auch dieses ein. Aber so, dass es vorkommen darf, nicht muss:

```
"Wetten(,)? da(ss|0337) ?\.{2,3} ?\?"
```

Hinter dem "{2,3}" ist einfach nur "?" dazugekommen. Damit sind wir auch schon fast fertig. Wenn da nicht hin und wieder eine Schreibweise auftauchen würde, bei der die Punkte zwischen dem „da“ und dem Fragezeichen fehlen.

Schauen Sie sich in der letzten Version genau an, welcher Teil für die zwei bis drei Punkte zuständig ist (incl. dem Leerzeichen dahinter). Genau dieser Teil wird in Klammern gesetzt und mit einem Fragezeichen alternativ gemacht:

```
"Wetten(,)? da(ss|0337) ?(\.{2,3} ?)\?"
```

Zum Abschluss, quasi als kleine Fingerübung, sorgen wir noch dafür, dass der gesuchte Begriff alleine in einer Zeile stehen muss:

```
"^Wetten(,)? da(ss|0337) ?(\.{2,3} ?)\??"
```

Es ist am Anfang nur das "^" und am Ende ein "\$" hinzugekommen. Jetzt sind wir aber endgültig fertig, hoffentlich nicht mit den Nerven. ;-)

Das ganze sieht jetzt sehr kompliziert aus, ist es in dem Beispiel natürlich auch. Es soll aber nur aufgezeigt werden, dass so ziemlich jedes Problem mit der Schreibweise von Suchbegriffen gelöst werden kann. Im Normalfall wird man nur wenige Elemente der regulären Ausdrücke gleichzeitig verwenden. Aber es ist ein beruhigendes Gefühl zu wissen, dass man zur Not noch viel mehr machen könnte.

In diesem Sinne:

Viel Spaß beim Suchen!

## 5. ASCII-Tabelle

Da u. U. nicht alle Zeichen auf der Tastatur verfügbar sind, kann man ein Zeichen auch durch seine Nummer angeben. Allerdings als Oktalzahl, die sich wie folgt zusammensetzt: "**\0**" und 3 Ziffern von 0-7, also z.B. "**\0334**" = "Ü".

Diese Oktalzahlen können überall verwendet werden, also in Zeichenketten, als Einzelzeichen oder als Zeichen in einer Zeichenklasse.

Dezimal	Oktal in iTiNa	Zeichen
32	\0040	Leerzeichen
33	\0041	!
34	\0042	"
35	\0043	#
36	\0044	\$
37	\0045	%
38	\0046	&
39	\0047	'
40	\0050	(
41	\0051	)
42	\0052	*
43	\0053	+
44	\0054	,
45	\0055	-
46	\0056	.
47	\0057	/
48	\0060	0
49	\0061	1
50	\0062	2
51	\0063	3
52	\0064	4
53	\0065	5
54	\0066	6
55	\0067	7
56	\0070	8
57	\0071	9
58	\0072	:
59	\0073	;
60	\0074	<
61	\0075	=
62	\0076	>
63	\0077	?
64	\0100	@
65	\0101	A
66	\0102	B
67	\0103	C
68	\0104	D
69	\0105	E
70	\0106	F
71	\0107	G
72	\0110	H
73	\0111	I
74	\0112	J
75	\0113	K

Dezimal	Oktal in iTiNa	Zeichen
76	\0114	L
77	\0115	M
78	\0116	N
79	\0117	O
80	\0120	P
81	\0121	Q
82	\0122	R
83	\0123	S
84	\0124	T
85	\0125	U
86	\0126	V
87	\0127	W
88	\0130	X
98	\0142	b
99	\0143	c
100	\0144	d
101	\0145	e
102	\0146	f
103	\0147	g
104	\0150	h
105	\0151	i
106	\0152	j
107	\0153	k
108	\0154	l
109	\0155	m
110	\0156	n
111	\0157	o
112	\0160	p
113	\0161	q
114	\0162	r
115	\0163	s
116	\0164	t
117	\0165	u
118	\0166	v
119	\0167	w
120	\0170	x
121	\0171	y
122	\0172	z
123	\0173	{
124	\0174	
125	\0175	}
126	\0176	~
127	\0177	□
128	\0200	€

Dezimal	Oktal in iTiNa	Zeichen
129	\0201	•
130	\0202	,
131	\0203	f
132	\0204	„
133	\0205	...
134	\0206	†
135	\0207	‡
136	\0210	^
137	\0211	‰
138	\0212	Š
139	\0213	‹
140	\0214	Œ
141	\0215	•
142	\0216	Ž
143	\0217	•
144	\0220	•
145	\0221	‘
146	\0222	’
147	\0223	“
148	\0224	”
149	\0225	•
150	\0226	—
151	\0227	—
152	\0230	~
153	\0231	™
154	\0232	š
155	\0233	›
156	\0234	œ
157	\0235	•
158	\0236	ž
159	\0237	ÿ
160	\0240	
161	\0241	ı
162	\0242	ç
163	\0243	£
164	\0244	¤
165	\0245	¥
166	\0246	ı
167	\0247	§
168	\0250	¨
169	\0251	©
170	\0252	ª
171	\0253	«
172	\0254	¬
173	\0255	—
174	\0256	®
175	\0257	-
176	\0260	°
177	\0261	±
178	\0262	²

Dezimal	Oktal in iTiNa	Zeichen
179	\0263	³
180	\0264	´
181	\0265	µ
182	\0266	¶
183	\0267	·
184	\0270	¸
185	\0271	¹
186	\0272	º
187	\0273	»
188	\0274	¼
189	\0275	½
190	\0276	¾
191	\0277	¿
192	\0300	À
193	\0301	Á
194	\0302	Â
195	\0303	Ã
196	\0304	Ä
197	\0305	Å
198	\0306	Æ
199	\0307	Ç
200	\0310	È
201	\0311	É
202	\0312	Ê
203	\0313	Ë
204	\0314	Ì
205	\0315	Í
206	\0316	Î
207	\0317	Ï
208	\0320	Ð
209	\0321	Ñ
210	\0322	Ò
211	\0323	Ó
212	\0324	Ô
213	\0325	Õ
214	\0326	Ö
215	\0327	×
216	\0330	Ø
217	\0331	Ù
218	\0332	Ú
219	\0333	Û
220	\0334	Ü
221	\0335	Ý
222	\0336	Þ
223	\0337	ß
224	\0340	à
225	\0341	á
226	\0342	â
227	\0343	ã
228	\0344	ä

Dezimal	Oktal in iTiNa	Zeichen
229	\0345	å
230	\0346	æ
231	\0347	ç
232	\0350	è
233	\0351	é
234	\0352	ê
235	\0353	ë
236	\0354	ì
237	\0355	í
238	\0356	î
239	\0357	ï
240	\0360	ð
241	\0361	ñ
242	\0362	ò

Dezimal	Oktal in iTiNa	Zeichen
243	\0363	ó
244	\0364	ô
245	\0365	õ
246	\0366	ö
247	\0367	÷
248	\0370	ø
249	\0371	ù
250	\0372	ú
251	\0373	û
252	\0374	ü
253	\0375	ý
254	\0376	þ
255	\0377	ÿ