Schnell-Filtersystem für Aktien

- Einleitung und Nutzung Aktien-Filter
- Beispiel-Filter
- Indikatoren-Funktionen in Filtern nutzen
- Reguläre Expressions für Text-Stammdaten

Einleitung und Nutzung Aktien-Filter

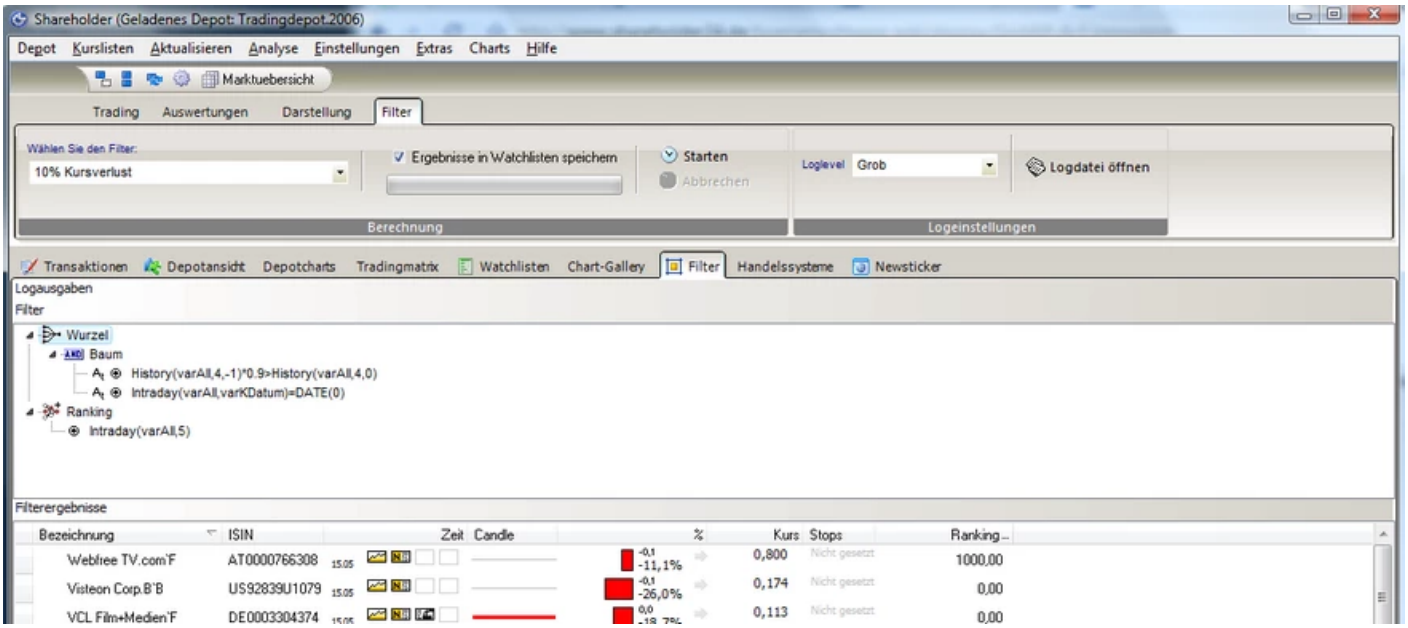
Zielsetzung

Der Filter in SHAREholder ist ein Filtersystem, dass die Eingrenzung von Werten auf Basis von Kriterien erlaubt. Als Kriterien sind alle Stammdaten & Technische-Indikatoren in beliebigen Kombinationen umsetzbar. Die Ergebnisliste wird in speziellen Beobachtungslisten zur weiteren Bearbeitung gespeichert. Die Filter sollen somit die Screening und das Scannen von interessanten Titeln am Markt unterstützen. Eine Kauf/Verkaufsentscheidung muss aber vom Anleger selbst getroffen werden.

Zusammenspiel mit Handelssystem

Der Filter wird dabei bewertet anhand eines Handelssystems, was eine Beurteilung eines Filtersystems ja erst zulässt. Ein definierter Filterbaum wird dabei im historischen Kurskontext auf seine Zuverlässigkeit und seine Performance geprüft. Die Kriterien des Handelssystem z.B. wann Titel der Ergebnisliste eines Filter tatsächlich gekauft werden und wieder verkauft werden, werden zusammen mit einem Filter definiert. Durch die Bewertung eines Filtersystems ist auch deren zukünftige Optimierung mittels Genetischen Algorithmen möglich. Die notwendige Voraussetzung der Bewertbarkeit einer "Population/Individuum" ist durch die Verwendung des Handelssystems gegeben.

Aufbereitung und Nutzung



Die Darstellung und Nutzung der eingerichteten Filter findet sich unter dem Hauptreiter "Filter". Um einen Filter zu starten müssen folgende Schritte gemacht werden:

1. Selektion des gewünschten Filters in der Toolbar oben
2. Entscheidung ob die Ergebnisse des Filters als gleichnamige *Watchliste abgespeichert werden sollen. Ein Häkchen in der Toolbar hierfür ist ausreichend
3. Start mit Button "Starten"

Nach Auswahl erfolgt die Berechnung und Darstellung der gefundenen Werte im unteren Bereich. Zur Verbesserung der Übersicht wird auch immer der zugehörige Funktionsbaum dargestellt. Änderungen am Filter können jederzeit über Aufruf der Filter-Einstellungs-Masken mit der Funktionstaste <F12> oder übers Hauptmenü Analyse/Filter-System aufgerufen werden.

Nach der Berechnung der Signale/Filter werden alle gefundenen Werte gelistet und mit folgenden Werten dargestellt:

- Bezeichnung/WKN/Zeit/Candle/Kurs/Stops
- Ranking (Bewertungsfunktion im DynFilter)

Die Darstellung erfolgt in 3 Abschnitten

1. Logausgaben
2. Filter bzw. den Bedingungsbaum
3. Filterergebnisse

Jedes der Elemente kann durch Doppelklick auf die Titelleiste ein- und ausgeblendet werden. Hier im Beispiel sind die Logausgaben ausgeblendet.

Loglevel

Grundsätzlich werden alle Logausgaben in einer Logdatei mitgeschrieben. Diese kann jederzeit über die Toolbar mit "Logdatei öffnen" geöffnet werden. Parallel steht direkt im Frontend ein Live-Log zur Verfügung der während der Bearbeitung eines Filters alle Ausgaben anzeigt. Welchen Detailgrad die Ausgaben haben wird über den Log-Level in der Toolbar festgelegt.

Ausgangsüberlegungen

Ein Filter setzt sich aus den Bedingungen und den in einer Bedingung genutzten Vergleichswerten inkl.- Funktionen zusammen. Jede Einzelbedingung muss jedoch immer ein Wahrheitswert ergeben, d.h. Aussagen von wahr oder falsch ergeben. In einem Bedingungsbaum können die Einzelbedingungen mit UND, ODER, UND NICHT, Exklusives ODER (genau eine der Aussage muss wahr sein) verknüpft werden. Als Suchbasis dienen alle verfügbaren Papiere. Das Filtersystem zeigt sich somit komplett unabhängig von vorherigen Filterkriterien z.B. von Watchlisten oder Markteinstellungen und zwingt somit den User jedes verwendete Kriterien zu definieren und genau darüber nachzudenken.

Allgemeine Voraussetzungen:

- Das System soll den Semiprofessionalen Anleger bei der Anlageentscheidung unterstützen.
- Der Erfolg des Systems setzt eine Disziplin des Anwenders voraus insb. bei der Ausführung von Verkaufsorders, da sonst das System nicht nachvollziehbar validierbar ist.

Weitere Ausgangsvoraussetzungen für die Nutzung in einem Handelssystem

- Die Strategie baut auf eine Form des Performance-Tradings auf, d.h. Ziel ist es unter Abzug von Transaktionskosten und Slippageverlusten einen maximalen Gewinn zu erzielen .
- Es können, da hauptsächlich am Europäischen Markt gehandelt werden soll, nur Long-Positionen eingegangen werden. Shortpositionen werden nicht berücksichtigt bzw.

berechnet.
Nach der Erzeugung eines Kaufsignals durch das Handelssystem ist die Position als Wahlentscheidung zwischen:

- Eingehen der Position auf Basis der Schluss Auktion (Close der gleichen Periode)
- Eingehen der Position auf Basis des Eröffnungskurses des nächsten Tages (Open)
- Eingehen der Positon auf Basis des Anfangshandels, wenn das Signal bestätigt wird, d.h. der Eröffnungskurs liegt über dem Schlusskurs des Vortages. Die Realisierung kann mittels Stop-Buy begrenzt auf den entsprechenden Handelstag erfolgen.
  Die Realisierung des Handelssystems setzt kein Vollzeitaktivität des Anwenders voraus, verlangt aber eine Prüfung der Stoppkurse (Trailing-Stops) auf noch nicht geschlossene Systeme kurz vor bzw. nach Handelsende (je nach Basis: Open/Close)
- Ein Engagement des Anlegers darf nur erfolgen, wenn Möglichkeiten zum Verkauf auf Tagesbasis gegeben sind, d.h. im Urlaub sollten keine Positionen ohne gesetztem Stop eingegangen werden. Hoffnungsansätze sind verboten, d.h. das System muss Anwendung finden (können).

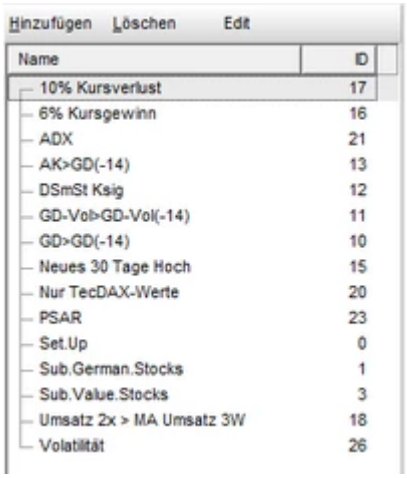
Außerhalb dieser Überlegung gibt es bereits ein Produkt das diese Teilmenge der Problematik bereits sehr gut löst, allerdings sehr stark im professionalen Segment (Preissegment) angesiedelt ist. Hier können über das bisher vorgeschlagene Konzept hinaus auch gezielt Future Trades, Put/Call-Scheine und Short-Positionen berücksichtigt werden. Die Kursdatenversorgung kann zudem mit fast allen gängigen Kursanbietern erfolgen.

Warum soll ein neues System bereitgestellt werden, wenn es bereits ein Produkt gibt, was die Anforderungen weitgehend erfüllen kann? Zum einen ist hier die Preispolitik ein entscheidender Faktor. Ebenso sind insb. die Candlestick-Formationen und eine Reihe von Indikatoren nicht im entsprechenden Maße abgedeckt, wie ich es mir vorstelle; ebenso wie eine Reihe von Unternehmensdaten. Dies ist jedoch meine subjektive Meinung, da es durchaus für einige Anleger entsprechend Gegenargumente geben kann.

Arbeitsschritte

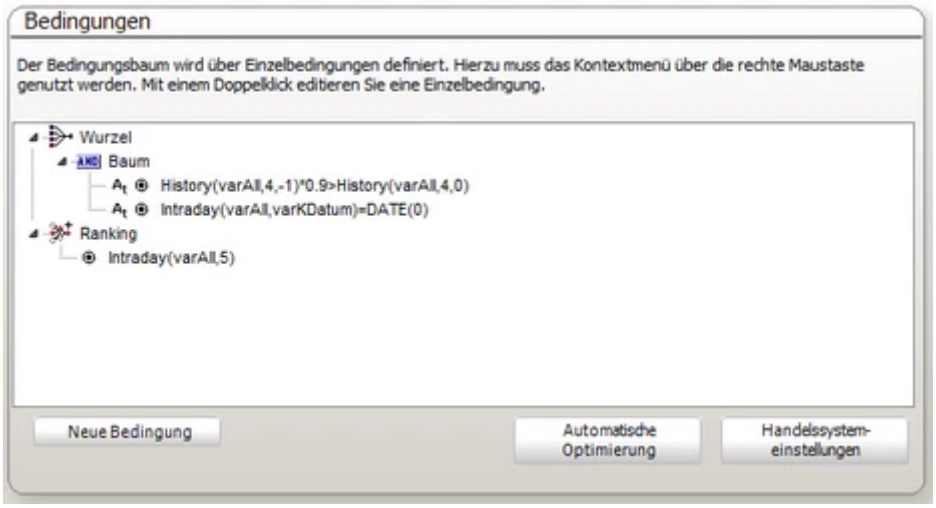
Die Bearbeitung von Filtern erfolgt unter Analyse / Filter-System. Der Shortcut hierzu ist <F12>. Alle folgenden Hinweise beziehen sich auf die Masken.

Anlegen eines Filters



In SHAREholder können beliebig viele Filter angelegt werden. Um einen neuen Filter anzulegen, muss mit dem Button "Hinzufügen" ein neuer Grundeintrag erzeugt werden. Es wird dabei automatisch die Frage gestellt, ob die Bedingungen aus dem aktuell selektierten Knoten übernommen werden sollen. Damit lassen sich dann leicht auch Kopien anfertigen.

Bearbeiten der Bedingungen eines Filters (Bedingungsbaum)



Jeder Filter besitzt einen Entscheidungsbaum, der alle Bedingungen in Beziehung zueinander setzt. Dabei sind als Verknüpfungstypen grundsätzlich

- UND
- ODER
- NOT (darf nicht)
- XOR (Exklusives ODER und somit muss genau 1 Werte wahr sein)
  erlaubt.

Der Bedingungsbaum verknüpft damit logisch Einzelbedingungen miteinander. Dies erfolgt in einer Baumansicht. So wird semantisch immer eine Verknüpfungstyp wie "UND" / "ODER" als Vater definiert und darunter die zu verknüpfenden Bedingungen z.B.

- AND
  - 1. Bedingung
  - 2. Bedingung
  - n. Bedingung

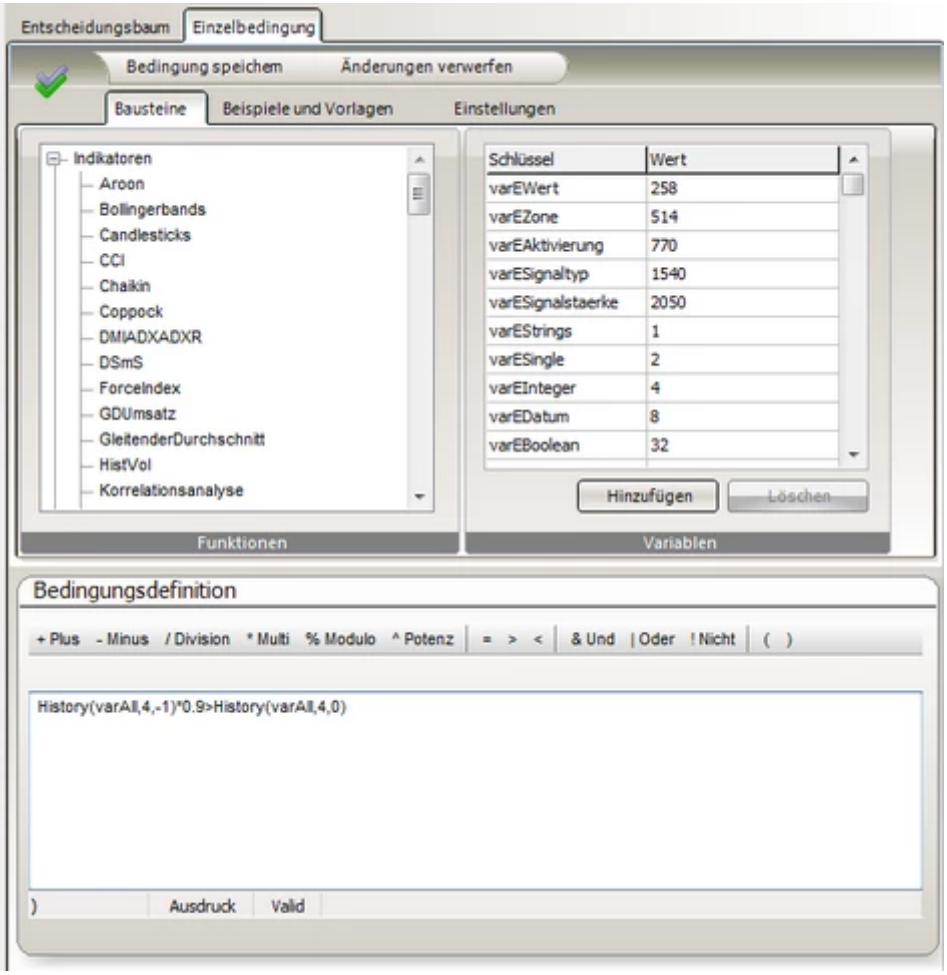
Die grundsätzliche Verarbeitung erfolgt dabei von oben nach unten. Die Reihenfolge ist damit entscheidend für die Verarbeitungsgeschwindigkeit eines Filters. Die Reihenfolge wird deshalb automatisch mit jedem vollständigen Suchvorgang optimiert anhand der statistischen Wahrscheinlichkeit $P^{(Fehler)}$. Dabei gilt, dass Bedingungen mit hoher Ausfallwahrscheinlichkeit an den Anfang eines Zweiges gesetzt werden bei UND - Bedingungen, bei OR entsprechend die Bedingungen die eine hohe Trefferwahrscheinlichkeit haben. Somit erfolgt die Präferenzbildung immer auf Basis einer möglichst schnellen Abwicklung des Zweiges festgelegt durch den Verknüpfungstyp .

Da jeder Filter durchaus mehrere Ergebniswerte zurückliefern kann, wird bei jedem Filter eine Rankingfunktion gesetzt, die die Sortierung der Ergebnisse erlaubt. Sie wird damit zu einer Bewertungsfunktion und führt zu einem Ranking innerhalb der Liste. Es kann pro Filter dabei immer nur eine Rankingfunktion geben. Müssen zwei verschiedene Funktionswerte genutzt werden, so muss ein mathematischer sinnvoller Ausdruck gefunden werden, der beide verknüpft.

Bedeutung hat dies entweder innerhalb einer persönlichen Rankingfunktion (manuellen) aber auch einer objektiven bei der Nutzung innerhalb eines Handelssystems. Hier werden die Positionen in der Ranking-Reihenfolge eingegangen, solange ausreichend Handelskapital zum Entscheidungstag zur Verfügung steht entsprechend den Einstellungen für die max. Kapitalsumme pro Position. Als Beispiel:

1. 1. Position entsprechend 1. Rankingposition mit 40% Kapitalsumme
2. 2. Position entsprechend 2. Rankingposition mit 40% Kapitalsumme
3. 3. Position wird nicht mehr eingegangen, da die Mindestanalagesumme nicht mehr zum Handelstag zur Verfügung steht.

Anlegen und Editieren einer Bedingung



Das Editieren einer Einzelbedingung erfolgt ausgehend vom Entscheidungsbaum im Editor "Einzelbedingung". Dieser arbeitet mit einer Autosuchfunktion, womit der User beliebige Variablen oder Funktionen eintippen kann und das System zeigt noch alle passenden Werte in einer Baumstruktur (Funktion) bzw. Listenansicht (Variablen) an. Die Validierung und Prüfung der Gültigkeit der aktuellen Eingabe wird fortwährend vom System vorgenommen. Grundsätzlich unterscheidet das System praktisch nur zwischen Mathematischen Ausdrücken und Patterns. Patterns meinen dabei beliebige Zeichenkettenvergleiche verbunden mit Oder | und Und & Vergleichen z.B. Markt='Ne*'|Markt='Nasdaq*'. Erlaubt eine Funktion Parameter, so werden diese in einem separaten Fenster abgefragt und dann als Ausdruck im Editor hinterlegt/angezeigt.

Die Einzelbedingungen werden vom System automatisch optimiert in ihrer Reihenfolge hinterlegt bei jeder Nutzung des Filters, d.h. Aussschluss-Kriterien mit hoher Wahrscheinlichkeit bei den bisherigen Prüfungen werden zuerst berücksichtigt, so d. alle nachfolgenden Berechnungen/Prüfungen nicht mehr notwendig werden.

Eine Einzeloptimierung der Parameter von Funktionen insb. der Technischen Indikatoren ist vorgesehen, jedoch noch nicht realisiert, da hierzu das Handelssystem erst fertiggestellt und optimiert werden muss.

Bedingungen sind entweder in mathematischer Form oder als Pattern zu formulieren. Hierbei entscheidet das Programm automatisch, wann welche Form genutzt wird, anhand der bisher genutzten Ausdrücke, Funktionen oder Variablen.

Verwendbare Filterkriterien bzw. Funktionen

Datenfelder und Kennzahlen	Weitere Details
Aktienstammdaten / Unternehmens-Stammdaten und Unternehmens-Kennzahlen	siehe Reguläre Expressions für Text-Stammdaten wie Notizen, Name, Markt etc. Alle in SHAREholder bekannten Kennzahlen sind auch im Filter nutzbar. • KGV(Basisjahr) : Kurs Gewinn Verhältnis des Basisjahres (1-nächste,0-aktuelle) • KBV : Kurs Buchverhältnis • KCV : Kurs Cashflow - Verhältnis • UMK : Umsatz Kursverhältnis • Buchwert • Cashflow • Aktienstueckzahl • Umsatz • Gewinn(Basisjahr) : Gewinn des Basisjahres (1-nächste,0-aktuelle) • Dividende(Basisjahr): Dividende des Basisjahres (1-nächste,0-aktuelle)e • PEG(Basisjahr) : Gewinnwachstum des Basisjahres (1-nächste,0-aktuelle) • DivR: Dividendenrendite
Technische Indikatoren und dessen Kauf-Verkaufssignale	siehe Indikatoren-Funktionen in Filtern nutzen

Datenfelder und Kennzahlen	Weitere Details
Mathematische und Logische Basisfunktionen	Einige mathematische und logische Basisfunktionen: • SQR(x) Quadrat X • SIN(x) Die Funktion Sin berechnet den Sinus eines Winkels. • COS(x) Die Funktion Cos berechnet den Cosinus eines Winkels im Bogenmaß • ATAN(x) Die Funktion ArcTan berechnet den Arcustangens einer bestimmten Zahl. • SINH(x) Die Funktion ArcSin berechnet den inversen Sinus einer bestimmten Zahl. • COSH(x) ArcCos berechnet den inversen Cosinus einer bestimmten Zahl. • COTAN(x) Die Funktion Cotan berechnet den Cotangens eines Winkels (Intern wird 1 / Tan(X) gerechnet) • TAN(x) Tan berechnet den Tangens von X. • EXP(x) Die Funktion Exp gibt die Potenz von X zurück. • LN(x) Die Funktion Ln gibt den natürlichen Logarithmus eines Real-Ausdrucks zurück (Ln(e) = 1) • LOG(x) Logarithmus zur Basis 10 • SQRT(x) Wurzel x • ABS(x) Die Funktion Abs gibt einen absoluten Wert zurück. • SIGN(x) Vorzeichen (-1,1,0) • TRUNC(x) Abschneiden der Nachkommastellen • CEIL(x) Ceil rundet den Wert einer Variablen auf z.B. Ceil(-2.8) = -2; Ceil(2.8) = 3; Ceil(-1.0) = -1 • FLOOR(x) Floor rundet Variablen ab z.B. Floor(-2.8) = -3; Floor(2.8) = 2; Floor(-1.0) = -1 • RND() Die Funktion Random erzeugt eine Zufallszahl innerhalb eines bestimmten Bereichs. • RANDOM() Initlisierung für die Nutzung von Zufallszahlen • INTPOW(x,y) IntPower errechnet die Potenz aus einer Basis. • POW(x,y) Power errechnet aus der Basis Base und dem beliebigen Wert Exponent die Potenz. • LOGN(x,y) LogN berechnet den Logarithmus X zur angegebenen Basis. • MIN(x,y) Minium von X, Y • MAX(x,y) Maximum von X,Y • IF(A,B,C) Wenn A>0 dann B sonst C

Datenfelder und Kennzahlen	Weitere Details
Kursdaten	Es gibt für den Zugriff auf Kursdaten zwei relevante Funktionen: • **Intraday**(<baiswert>,<kurstyp>) • **History**(<basiswert>,<kurstyp>,<Heute-X-Zeitpunkt>) Der Basiswert (Bezugswert für die Auswertung) ist im Normalfall mit "varAll" belegt und somit innerhalb einer Suchliste variabel, d.h. es wird immer der aktuelle Titel als Basiswert gesetzt und kein fixer Titel. Es ist aber auch möglich z.B. den TecDAX mit seiner (!)WKN zu verwenden "Intraday(<TECDAX-WKN>,<KursTyp>)" und hier einzusetzen. Es ist somit der betrachtete Wert unbestimmt mit varAll oder bereits vorbelegt mit der WKN nutzbar. Das Prinzip ist so für alle vorhandenen Funktionen übertragbar. Als Kurstypen (<kurstyp>) sind erlaubt (O-1,H-2,L-3,C-4,V-5,D-8). Um die Schreibweise leserlicher zu machen (statt der Zahlenwerte), gibt es korrespendierende Konstanten: varKOpen, varKHigh, varKLow, varKClose, varKVolumen, varKDatum. Um die Lesbarkeit zu behalten, sollten diese wenn möglich auch immer verwendet werden. Diese werden aber nicht erzwungen. Intraday gibt dabei immer den aktuellen interen Tageskurs zurück. Liegen noch keine aktuellen Daten vor, wird der letzte bekannte Wert zurückgegeben. Der Wert entspricht immer der Anzeige in den Kurslisten unter AK (Aktuellem Kurs). Heute-X-Zeitpunkt für History History besitzt ein Parameter der bei 0 beginnt zu zählen, d.h. mit 0 : Tageswert und somit = Intraday entsprechend dem gewählten Kurstyp z.B. • History(varAll,varKClose,0) - Heutiger Schlusskurs bzw. letzter AK von heute. • History(varAll,varKClose,-1) - Vortageswert • History(varAll,varKClose,-2) - Vorgestern usw. Logisch ist damit • History(<basiswert>,<kurstyp>,0) = Intraday(<basiswert>,<kurstyp>) Ebenfalls verwendet werden kann für die Rückgabe von Kursdaten. Hierbei kann jedoch auf keinen Fremdtitel referenziert werden, sondern es wird im Filter immer der jeweilige aktuell geprüfte Titel verwendet im Filter-Durchlauf: **RasInt** und **RasHis**(O|H|L|C|V|Min|Max) wobei die Felder als Zeit oder Indexfelder verwendet werden können • z.B. RasHisC(-2) gibt den Schlusskurs von vor 2 Handelstagen wieder • z.B. RasHisMax(-5) gibt den Höchstkurs der letzten 5 Handelstage wieder **HistoryCount**() gibt die Anzahl der vorhandenen Kursdaten auf Schluss-Kurs-Basis zurück. Dies soll vor allem dem herausfiltern von Titeln dienen, die keine ausreichende Kursbasis haben.

Datenfelder und Kennzahlen	Weitere Details
Nutzung der Ergebnisse eines anderen Filters (SubFilter)	Die Bedingungen in einem DynFilter können zusätzlich strukturiert werden innerhalb von Sub-Filtern, d.h. es können definierte Filter in anderen Filtern aufgerufen werden. Dieser Aufruf darf nicht rekursiv erfolgen. Die Berechnung des Subfilters erfolgt zu dem Zeitpunkt der ersten Nutzung innerhalb eines Bedingungsbaumes, sofern dies logisch notwendig ist (wenn die vorherigen AND Bedingungen nicht erfüllt werden können, kommt es zu keinem Aufruf der Funktion). Die Ergebnisse des Filters werden zwischengespeichert, so dass ein nochmaliger Zugriff nicht zu einer kompletten Neuberechnung führt. Die Zugriffsfunktionen sind: • **SubFilter**(ID des Filters,varAll,Ergebniswert) Als Ergebniswerte eines Subfilters sind definiert: • varFilterRanking - Rückgabe des Rankingwertes des betrachteten Basistitels • varFilterExists - 1 wenn existiert • varFilterPos - Position in der Ergebnismenge auf Basis der sortierten Liste mittels der Rankingfunktion zulässig. Somit läßt sich sowohl die Position als auch die Existenz eines Wertes in einem anderen Filter ermitteln. Auch die Auswertung auf Basis eines bestehenden Rankings ist möglich.
Datumsfunktionen	• **toDate**(DD,MM,YYYY): EXTENTED • **toTime**(HH,MI): EXTENTED (Nachkommastelle) • **BaseDATE**(<basis-verschiebung>) : EXTENTED, • **FirstEOD**: EXTENDED, womit der erste Kurs des Bastitels ermitteln wird und ein Vergleich zur Prüfung einer ausreichenden Kursbasis möglich wird z.B. FirstEOD<toDate(01,01,2000) möglich wird wobei <basis-verschiebung> zwischen -3000 und 3000 liegen kann und sich auf das aktuell Datum bezieht

Optimierung

Zur Zeit gibt es nur eine Variante eines automatischen Brute-Force-Optimierungsalgorithmus, d.h. es wird der Filter komplett berechnet mit vollständiger Auswertung aller logischen Ausdrücke (auch wenn logisch bereits klar ist, dass der Gesamtausdruck festgelegt ist wahr/falsch). Somit werden die Laufzeiten und die Gesamt-Aufrufe der Einzelbedingungen festgestellt.

Jede Einzelbedingung wird bewertet mit

- Zeitbedarf x Aufrufe

Alle Einzelbedingungen können so sortiert werden nach logischem Ausschluss. Hierbei werden die Bedingungen, die ein sehr hohen "Aufwand" produzieren innerhalb eines Teilbaumes nach hinten sortiert. In der Praxis können so die weniger ressourcenhungrigen Bedingungen zuerst berechnet werden um dann (wenn noch notwendig) die weiteren Bedingungen auszuwerten.

In der Summe sollte so ein erstellter Filter immer durch das System optimiert werden. Eine manuelle Änderung der Reihenfolge der Einzelbedingungen innerhalb eines Teilbaumes ist zur Zeit vorgesehen, jedoch noch nicht integriert.

Bewertung eines Filters mit Handelssystem

siehe Handelssystem

Beispiel-Filter

Programminterne Beispiele

Sie finden direkt im Programm eine editierbare Beispielleiste. Die Beispiele können mit Doppelklick direkt übernommen werden. Man kann sich so schnell eine eigene Sammlung von sinnvollen Bedingungen zusammenstellen. Die Beispiele können mit Doppelklick als Ausdruck für eine Bedingung übernommen werden. Die Beispiele können selbst erweitert und ergänzt werden über eine editierbare Beispiel-Textdatei, die auf der Oberfläche auch angezeigt wird. Wichtig ist, dass ein Filter sich immer aus mehreren Einzelbedingungen zusammensetzen. Ich kann zwar auch komplizierte Bedingungen in ein Bedingungsausdruck miteinander verknüpfen; dies sollte man aber vermeiden.

Einfache Beispiele kurz erklärt

Nachfolgend einige formulierte Beispiele, die so auch in der Standardauslieferung enthalten sind:

Durchschnittsvolumen der letzten 5 Tage

Um das Durchschnittsvolumen von den letzten 5 Tagen zu bekommen wäre somit simpel zu schreiben:

```
(History(varAll,varKVolumen,-5)
+History(varAll,varKVolumen,-4)
+History(varAll,varKVolumen,-3)
+History(varAll,varKVolumen,-2)
+History(varAll,varKVolumen,-1))/5 < History(varAll,varKVolumen,0)
```

Mindestvolumen von 5000 Stk.

Wert hat an dem Tag ein bestimmtes mindest Volumen, z. B. 5000 Stk.

```
Intraday(varAll,varKVolumen)>5000
```

Kurs von über 0,05 Euro

```
Intraday(varAll,varKClose)>0,05
```

Kursgewinn von 15%

```
History(varAll,varKClose,0)>History(varAll,varKClose,-1)*1,15
```

4x so großes Durchschnittsvolumen

Wert hat einen 4x so großes Volumen als der Durchschnittswert der letzten 30 Tage.

```
(Intraday(varAll,varKVolumen)>GDUmsatz(varIGMittelfristigeEinstellungen,varAll,varEWert,0)*4) &
(GDUmsatz(varIGMittelfristigeEinstellungen,varAll,varEWert,0)>0)
```

Rein theoretisch könnte man auch GleitenderDurchschnitt als Funktion mit einer festgelegten Basisfestlegung auf den Kurstyp "Volumen" benutzen. Dies ist aber umständlich und erzwingt zudem eine eigene Indikatorengruppe, da diese Festlegung global gelten insb. für Charts usw.

Die 30 Tage verbergen sich hierbei in den Einstellungen für den Indikator "Gleitenden Durchschnitt" für die Indikatorengruppe "Mittelfristige Einstellungen".

Wert hat an dem Tag ein neues 60 Tage Hoch

```
NewHigh(varIGMittelfristigeEinstellungen,varAll,varEAktivierung,0)>0
```

Die 60 Tage müssen hierbei am Indikator eingetragen werden z.B. im Chart, im Indikatorenpannel, im Wizard oder auch einfach direkt unter Einstellungen.Indikatorengruppe ->Doppelklick (siehe Indikatorenfunktioen)

Nur deutsche Werte

```
Markt='*DAX*'|Markt='*Standard*'|Markt='Ne*'|Markt='Pri*'|Markt='General*'
```

Es gibt keine direkte Variante für diese Prüfung. Die Prüfung muss anhand der Marktdaten erfolgen. Hierbei werden alle relevanten Märkte in einem regulären Ausdruck vergleichen. Im Beispiel

erfüllen somit alle Werte aus Märkten mit dem Namen **DAX** oder **Standard** usw. die Bedingung. Der Ausdruck sollte aber einmal einfach definiert werden und dann über SubFilter(<ID des Filters>,varAll) eingebunden werden. Erspart hier und an anderen Stellen etwas Arbeit.

Indikatoren-Funktionen in Filtern nutzen

Nutzung von Indikatoren in Filtern

Wenn Funktionen Parameter erwarten, so wird automatisch eine Eingabehilfe aufgerufen, die hoffentlich selbsterklärend ist.
Es gibt dabei eine Reihe von nutzbaren Funktionen, wobei folgende Zuordnung genutzt wird:

Die Funktionen basieren immer auf einer Einstellungsgruppe, die unter im Hauptmenü im Hauptfenster unter Einstellungen / Indikatoren neu angelegt werden kann. Das Anpassen der Einstellungen kann dann direkt mit dem Button "Verändern" vorgenommen werden.

Es gilt immer folgende Semantik:
Funktionsname(<IndikatorengruppeID>,<Suchraum als WKN oder varAll>,<Ergebnistyp>,<Ergebnis-Zeit-Punkt relativ mit 0,-1,-2 oder als toDate()>

Element	Details
<Indikatorengruppe>	Für jede Indikatorengruppe wird immer eine Konstante angelegt z.B. varIGMittelfristigeEinstellungen mit varIG<Name>. Diese kann für die Referenzierung genutzt werden. Wichtig ist hierbei zu verstehen, dass spezifische Indikatorenparameter z.B. die Anzahl Tage für die Moving-Average-Berechnung Kurz und lang über die Indikatorengruppen gepflegt werden und nicht direkt als Berechnungsparameter mitgegeben werden. Dies erscheint im ersten Moment umständlich, vereinfacht aber Optimierungen im Chart (manuell), die sofort für die Filter gültig werden ohne zusätzliche Änderungen. Die Änderung der Werte zu einem Indikator kann auf sehr unterschiedlichem Wege erfolgen. Mit diesem Indikatoren-Gruppen-Konzept sind die Indikatorenparameter in den Charteinstellungen, Filterberechnungen, Handelssystemen immer gleich und können dennoch unterschieden werden.
<Basiswert>	Entweder <varAll> oder eine spezifische WKN eines Titels, was für die Berechnung verwendet werden soll

Element	Details
<Basisergebnis>	Als Funktionsergebniss ist folgendes erlaubt: • Ergebniswert(varEWert) - Als Ergebnis der Funktion wird der Wert der Funktion zum Tag X ausgegeben. • Zonenwert (varEZone) - Der Rückgabewert liegt zwischen 1 und 3 und nimmt als Basis das Funktionsergebnis. Die Einordnung in eine Zone erfolgt immer auf Basis der Einstellungen eines Indikators. Z.B. kann die Einordnung mittels Normalverteilung oder Gleichverteilung vorgenommen werden. Im Ergebnis werden die Werte über einem festgestellten oberen Schwellwert mit Zone=3 und Werte unterhalb eines errechneten Schwellwertes mit Zone=1 zurückgegeben. • Aktivierung (varEAktivierung) - Ähnlich des Prinzips der Fuzzy Logik, kann jeder errechnete Kauf/Verkaufszeitpunkt eines Indikators mit einer Verfallszeit eingestellt werden. Dies sorgt dafür, dass das Signal "ausschwingt" in den folgenden Tagen mittels einer tanh Funktion (1-tanh(2*Zeit/Signale.Verfallszeit)). Die Signale sind somit mit abnehmender Zeit bis zum Verfallstag immer deutlich schwächer. Verkaufssignale ergeben werden zwischen -1 und 0 und Kaufsignale ergeben immer Werte zwischen 0 und 1. Den Verlauf der Signalmuster kann eingestellt werden unter den Optionen eines Indikators und betrachtet werden in einem Chart unter Aktivierung der Signalmuster mit: • Signaltyp/ Signalstärke (varESignaltyp)- Hier werden generierte Kauf-und Verkaufssignale mit -1 und 1 (Kauf) zurückgegeben. Signaltyp und Signalstärke sind dabei synonym verwendet mit Ausnahme der Candlestickformationen. Candlestickformationen besitzen eine eigene Verwendung der Ergebnismengen mit: ◦ Zonen sind immer = 0 (es findet somit keine Berechnung statt) ◦ Aktivierung = Signaltyp = 1 bei Kauf und -1 bei Verkauf ◦ Wert = Typ der Formation entsprechend den definierten Variablen z.B. cCandleFHangingMan usw. ◦ SigStaerke = Signaltyp*Candlestickformationen
<Ergebnis-Zeit-Punkt relativ mit 0,-1,-2 oder als toDate()>	Das Ergebnis eines Indikators kann relativ betrachtet zurückgegeben werden d.h. z.B. -2 von vor 2 Handelstagen. Mittels Übergabe der toDate-Funktion ist auch ein absolutes Datum nutzbar.

Bei Ausführung des Filters sollte man bei Unstimmigkeiten oder zur Kontrolle den Detail-Log-Level aktiviert haben. Hier sind dann die Rückgabewerte der Funktionen prüfbar.

Besondere Indikatoren und dessen Nutzung

Neuronale Netze für Kursprognose

Die Neuronalen Prognose-Netze können in einem Filtersystem genutzt werden. Der Zugriff erfolgt immer über die NNPrognose - Funktion, unabhängig vom genutzter Netz.

- NNPrognose(ID des Netzes,Wert (z.B. valAll),Zielwert (z.B.varPrognoseHeute))

Bei der Nutzung eines solchen Ausdruckes, wird im Programm automatisch durch einen Assistenten die möglichen Parameter, IDs zusammengestellt.

Herleitung zusammen mit einem Chart-Indikator-Bild

unknown-attachment?locale=de_DE&version=2

Zielstellung

Das Ziel ist es einen Filter zu erstellen, der mir die Aktien heraussucht, die ADX > 30 über die letzten 14 Tage hatte. Die Herleitung sollte hier am einfachsten visuell erfolgen können. Die Einstellungen von Indikatoren werden immer in Indikatorengruppen gespeicher, d.h. möchte man ein und derselben Indikator auf unterschiedliche Zeiträume vergleichen, muss man entsprechende Indikatorengruppen bilden und diese im Filter entsprechend nutzen. Nachdem man einen neuen Filter erstellt hat und eine UND-Bedingung sowie eine einfache Bedingung im Baum angelegt hat, geht man per Doppelklick auf die neue Bedingung in den Editor. In diesem Bedingungseditor, kommt man durch Eingabe von "DMI" sofort in den Wizard.

Hier kann man nochmals die Parameter des Indikators erkennen und verändern (Button Verändern). Wichtig ist die ganz untere Zeile, die dem eigentlichen Funktionsaufruf bildet, d.h. es gilt folgende Semantik

Funktionsname(<IndikatorengruppeID>,<Suchraum als WKN oder varAll>,<Ergebnistyp>,<ErgebnisPunkt relativ mit 0,-1,-2 oder als toDate()>)

Wichtig ist hier vor allem den Ergebnistyp zu verstehen.

Diesen Ergebnistyp kann man gut im Chart vergleichen und erkennen. Wert resultiert hier aus der Indikator-Ergebnislinie (graue) und wird bei Mausbewegung auch direkt angezeigt mit Wert. Der Zonenwert liegt zwischen 1-3 und kann für bestimmte Auswertungen helfen, da die Zonenberechnung unterschiedliche Methoden erlaubt die über ein >MinWert und <MaxWert hinausgehen. Der Aktivierungsgrad liegt zwischen -1 und 1 und entspricht nachlaufenden Kauf-und Verkaufssignalen, d.h. wenn ich nicht umständlich prüfen will ob ein Indikator ein Kaufsignal in den letzten 5 Tagen gebildet hat (womit ich ja DMI(....,-1) | DMI(...,-2) usw. schreiben müsste kann ich

die Verfallszeit eines Signals einstellen z.B. auf 5 Tage. So länger der Wert her ist um so schwächer wird er. Dies ist im Chartbild auch gut zu erkennen.

Das Signal selber wird im Chart über rote und grüne Pfeile dargestellt. Die Signalumgebung kann erzeugt werden mit Richtungsänderung, Nullinienschnittpunkt usw. Da die Indikatoren unterschiedlich ausgelegt werden können, habe ich die Signalumgebung entsprechend flexibel ausgelegt. Der Signaltyp wird im Normalfall im 1 für Kauf und -1 für Verkauf sein. Für spezielle Indiaktoren wie die Candlestickformation sind zusätzliche Zuordnungen über Konstanten erlaubt. Die Signalstärke bei Candlestickformationen ergeben dabei statisch (siehe Hilfe für Candlesticks) festgelegte Rückgabewerte im Bereich 1-5.

Für Ihr Beispiel ergibt sich somit (14 Tage muss in der Indikatorengruppe eingestellt werden und ist damit auch automatisch für die Chartdarstellung gültig -> deshalb können die Indikatoren nicht direkt verändert werden)

```
DMIADXADXR(1,varAll,varEWert,0)>30
```

Etwas schöner geschrieben wäre auch

```
DMIADXADXR(varIGMittelfristigeEinstellungen,varAll,varEWert,0)>30
```

möglich.

Alle Indikatorengruppe werden dabei als Konstanten mit "varIG<Name>" abgelegt. Der Wizard arbeitet aber bewußt mit technischen Nummern, da diese weniger anfällig gegen Umbenennungen sind.

Reguläre Expressions für Text-Stammdaten

Grundprinzip

Für Strings werden ab der Version 13.2.8 werden normale reguläre Expressions genutzt. Damit sind grundsätzlich Zeichenauswahl-Filter (**[egh]**) , vordefinierte Zeichenklassen (\d - Zahl), Quantoren und auch die Behandlung von Sonderzeichen möglich für komplexere mehrzeilige Ausdrucks-Vergleiche.

Es ist lediglich zu beachten, dass zunächst immer die <Filterfunktion> getrennt durch das Tilde-Zeichen ~ und der regulären Expression definiert sein muss. Die Groß-und Kleinschreibung wird ignoriert.

Nutzbare Stammdaten

Folgende Stammdaten-Felder lassen sich in Filtern auswerten und nutzen.

Wert	Rückgabe-Beispiel	Semantik und Besonderheiten
WKN		<WKN>
ISIN oder Symbol		<ISIN>
Videotextname		
Markt	_DAX_,_MDAX_	analog Segments nur mit Kurznamen der Märkte und Verwendung von _
Notizen		Mehrzeiliger Text, der über den Chart oder über Doppelklick auf einen Titel gepflegt werden kann
Kurzname		
BasiswertISIN		

Wert	Rückgabe-Beispiel	Semantik und Besonderheiten
Variables	Yahoo-Symobl[isin]:LEO,	Zugeordnete und vorhandene Internet-Variablen z.B. Yahoo-Symbole <Variablenname>:<Wert>,
Segments	DAX,MDAX,Deutschland,	Zugeordnete <Marktsegment>,
Watchlists	Märkte,meineWatchlist,*Depot,	Zugeordnete <Watchlist>,

Verwendung von reguläre Expressions

Weitere Details zur Verwendung finden Sie hier:

https://www.regular-expressions.info/reference.html oder
https://de.wikipedia.org/wiki/Regul%C3%A4rer_Ausdruck

Regular Expression Basic Syntax Reference

Characters		
Character	**Description**	**Example**
Any character except [\^$.|?*+()	All characters except the listed special characters match a single instance of themselves. { and } are literal characters, unless they're part of a valid regular expression token (e.g. the {n} quantifier).	a matches a
\ (backslash) followed by any of [\^$.|?*+(){}	A backslash escapes special characters to suppress their special meaning.	\+ matches +
\Q...\E	Matches the characters between \Q and \E literally, suppressing the meaning of special characters.	\Q+-*/\E matches+-*/

\xFF where FF are 2 hexadecimal digits	Matches the character with the specified ASCII/ANSI value, which depends on the code page used. Can be used in character classes.	\xA9 matches © when using the Latin-1 code page.
\n, \r and \t	Match an LF character, CR character and a tab character respectively. Can be used in character classes.	\r\n matches a DOS/Windows CRLF line break.
\a, \e, \f and\v	Match a bell character (\x07), escape character (\x1B), form feed (\x0C) and vertical tab (\x0B) respectively. Can be used in character classes.	
\cA through \cZ	Match an ASCII character Control+A through Control+Z, equivalent to\x01 through \x1A. Can be used in character classes.	\cM\cJ matches a DOS/Windows CRLF line break.
Character Classes or Character Sets [abc]		
Character	**Description**	**Example**
[(opening square bracket)	Starts a character class. A character class matches a single character out of all the possibilities offered by the character class. Inside a character class, different rules apply. The rules in this section are only valid inside character classes. The rules outside this section are not valid in character classes, except for a few character escapes that are indicated with "can be used inside character classes".	
Any character except ^-]\ add that character to the possible matches for the character class.	All characters except the listed special characters.	[abc] matches a, b orc
\ (backslash) followed by any of ^-]\	A backslash escapes special characters to suppress their special meaning.	[\^\]] matches ^ or]
- (hyphen) except immediately after the opening [	Specifies a range of characters. (Specifies a hyphen if placed immediately after the opening [)	[a-zA-Z0-9] matches any letter or digit
^ (caret) immediately after the opening [	Negates the character class, causing it to match a single character*not* listed in the character class. (Specifies a caret if placed anywhere except after the opening [)	[^a-d] matches x (any character except a, b, c or d)
\d, \w and \s	Shorthand character classes matching digits, word characters (letters, digits, and underscores), and whitespace (spaces, tabs, and line breaks). Can be used inside and outside character classes.	[\d\s] matches a character that is a digit or whitespace

\D, \W and \S	Negated versions of the above. Should be used only outside character classes. (Can be used inside, but that is confusing.)	\D matches a character that is not a digit
[\b]	Inside a character class, \b is a backspace character.	[\b\t] matches a backspace or tab character
Dot		
Character	**Description**	**Example**
. (dot)	Matches any single character except line break characters \r and \n. Most regex flavors have an option to make the dot match line break characters too.	. matches x or (almost) any other character
Anchors		
Character	**Description**	**Example**
^ (caret)	Matches at the start of the string the regex pattern is applied to. Matches a position rather than a character. Most regex flavors have an option to make the caret match after line breaks (i.e. at the start of a line in a file) as well.	^. matches a inabc\ndef. Also matches d in "multi-line" mode.
$ (dollar)	Matches at the end of the string the regex pattern is applied to. Matches a position rather than a character. Most regex flavors have an option to make the dollar match before line breaks (i.e. at the end of a line in a file) as well. Also matches before the very last line break if the string ends with a line break.	.$ matches f inabc\ndef. Also matches c in "multi-line" mode.
\A	Matches at the start of the string the regex pattern is applied to. Matches a position rather than a character. Never matches after line breaks.	\A. matches a in abc
\Z	Matches at the end of the string the regex pattern is applied to. Matches a position rather than a character. Never matches before line breaks, except for the very last line break if the string ends with a line break.	.\Z matches f inabc\ndef
\z	Matches at the end of the string the regex pattern is applied to. Matches a position rather than a character. Never matches before line breaks.	.\z matches f inabc\ndef
Word Boundaries		
Character	**Description**	**Example**

\b	Matches at the position between a word character (anything matched by \w) and a non-word character (anything matched by [^\w] or \W) as well as at the start and/or end of the string if the first and/or last characters in the string are word characters.	.\b matches c in abc
\B	Matches at the position between two word characters (i.e the position between \w\w) as well as at the position between two non-word characters (i.e. \W\W).	\B.\B matches b in abc
Alternation		
Character	**Description**	**Example**
| (pipe)	Causes the regex engine to match either the part on the left side, or the part on the right side. Can be strung together into a series of options.	abc|def|xyz matchesabc, def or xyz
| (pipe)	The pipe has the lowest precedence of all operators. Use grouping to alternate only part of the regular expression.	abc(def|xyz)matches abcdef orabcxyz
Quantifiers		
Character	**Description**	**Example**
? (question mark)	Makes the preceding item optional. Greedy, so the optional item is included in the match if possible.	abc? matches ab orabc
??	Makes the preceding item optional. Lazy, so the optional item is excluded in the match if possible. This construct is often excluded from documentation because of its limited use.	abc?? matches ab orabc
* (star)	Repeats the previous item zero or more times. Greedy, so as many items as possible will be matched before trying permutations with less matches of the preceding item, up to the point where the preceding item is not matched at all.	".*" matches"def" "ghi" inabc "def" "ghi" jkl
*? (lazy star)	Repeats the previous item zero or more times. Lazy, so the engine first attempts to skip the previous item, before trying permutations with ever increasing matches of the preceding item.	".*?" matches "def"inabc "def" "ghi" jkl

+ (plus)	Repeats the previous item once or more. Greedy, so as many items as possible will be matched before trying permutations with less matches of the preceding item, up to the point where the preceding item is matched only once.	".+" matches"def" "ghi" inabc "def" "ghi" jkl
+? (lazy plus)	Repeats the previous item once or more. Lazy, so the engine first matches the previous item only once, before trying permutations with ever increasing matches of the preceding item.	".+?" matches "def"inabc "def" "ghi" jkl
{n} where n is an integer >= 1	Repeats the previous item exactly n times.	a{3} matches aaa
{n,m} where n >= 0 and m >= n	Repeats the previous item between n and m times. Greedy, so repeating m times is tried before reducing the repetition to n times.	a{2,4} matches aaaa,aaa or aa
{n,m}? where n >= 0 and m >= n	Repeats the previous item between n and m times. Lazy, so repeating n times is tried before increasing the repetition to m times.	a{2,4}? matches aa,aaa or aaaa
{n,} where n >= 0	Repeats the previous item at least n times. Greedy, so as many items as possible will be matched before trying permutations with less matches of the preceding item, up to the point where the preceding item is matched only n times.	a{2,} matches aaaaain aaaaa
{n,}? where n >= 0	Repeats the previous item n or more times. Lazy, so the engine first matches the previous item n times, before trying permutations with ever increasing matches of the preceding item.	a{2,}? matches aa inaaaaa

Regular Expression Advanced Syntax Reference

Grouping and Backreferences		
Syntax	**Description**	**Example**

(regex)	Round brackets group the regex between them. They capture the text matched by the regex inside them that can be reused in a backreference, and they allow you to apply regex operators to the entire grouped regex.	(abc){3}matchesabcabcabc. First group matches abc.
(?:regex)	Non-capturing parentheses group the regex so you can apply regex operators, but do not capture anything and do not create backreferences.	(?:abc){3}matchesabcabcabc. No groups.
\1 through \9	Substituted with the text matched between the 1st through 9th pair of capturing parentheses. Some regex flavors allow more than 9 backreferences.	(abc|def)=\1matchesabc=abc ordef=def, but not abc=def ordef=abc.
Modifiers		
Syntax	**Description**	**Example**
(?i)	Turn on case insensitivity for the remainder of the regular expression. (Older regex flavors may turn it on for the entire regex.)	te(?i)stmatches teSTbut not TEST.
(?-i)	Turn off case insensitivity for the remainder of the regular expression.	(?i)te(?-i)stmatches TEstbut not TEST.
(?s)	Turn on "dot matches newline" for the remainder of the regular expression. (Older regex flavors may turn it on for the entire regex.)	
(?-s)	Turn off "dot matches newline" for the remainder of the regular expression.	
(?m)	Caret and dollar match after and before newlines for the remainder of the regular expression. (Older regex flavors may apply this to the entire regex.)	
(?-m)	Caret and dollar only match at the start and end of the string for the remainder of the regular expression.	
(?x)	Turn on free-spacing mode to ignore whitespace between regex tokens, and allow # comments.	
(?-x)	Turn off free-spacing mode.	
(?i-sm)	Turns on the option "i" and turns off "s" and "m" for the remainder of the regular expression. (Older regex flavors may apply this to the entire regex.)	

(?i-sm:regex)	Matches the regex inside the span with the option "i" turned on and "m" and "s" turned off.	(?i:te)stmatches TEstbut not TEST.
Atomic Grouping and Possessive Quantifiers		
Syntax	**Description**	**Example**
(?>regex)	Atomic groups prevent the regex engine from backtracking back into the group (forcing the group to discard part of its match) after a match has been found for the group. Backtracking can occur inside the group before it has matched completely, and the engine can backtrack past the entire group, discarding its match entirely. Eliminating needless backtracking provides a speed increase. Atomic grouping is often indispensable when nesting quantifiers to prevent a catastrophic amount of backtracking as the engine needlessly tries pointless permutations of the nested quantifiers.	x(?>\w+)x is more efficient than x\w+x if the second x cannot be matched.
?+, *+, ++ and{m,n}+	Possessive quantifiers are a limited yet syntactically cleaner alternative to atomic grouping. Only available in a few regex flavors. They behave as normal greedy quantifiers, except that they will not give up part of their match for backtracking.	x++ is identical to (?>x+)
Lookaround		
Syntax	**Description**	**Example**
(?=regex)	Zero-width positive lookahead. Matches at a position where the pattern inside the lookahead can be matched. Matches only the position. It does not consume any characters or expand the match. In a pattern likeone(?=two)three, both two and three have to match at the position where the match of one ends.	t(?=s)matches the second t instreets.
(?!regex)	Zero-width negative lookahead. Identical to positive lookahead, except that the overall match will only succeed if the regex inside the lookahead fails to match.	t(?!s)matches the firstt in streets.

(?<=regex)	Zero-width positive lookbehind. Matches at a position if the pattern inside the lookahead can be matched ending at that position (i.e. to the left of that position). Depending on the regex flavor you're using, you may not be able to use quantifiers and/or alternation inside lookbehind.	(?<=s)tmatches the firstt in streets.
(?<!regex)	Zero-width negative lookbehind. Matches at a position if the pattern inside the lookahead cannot be matched ending at that position.	(?<!s)tmatches the second t instreets.

Continuing from The Previous Match

Syntax	Description	Example
\G	Matches at the position where the previous match ended, or the position where the current match attempt started (depending on the tool or regex flavor). Matches at the start of the string during the first match attempt.	\G[a-z] first matches a, then matches b and then fails to match in ab_cd.

Conditionals

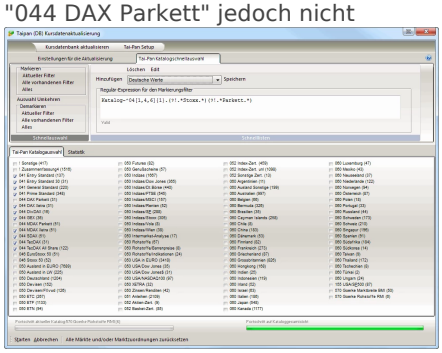
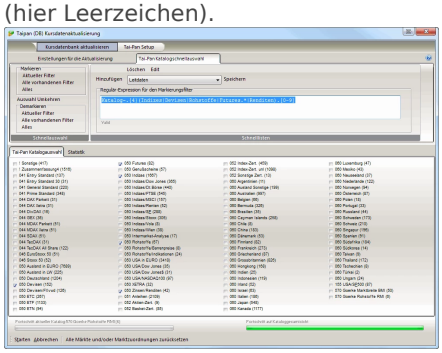
Syntax	Description	Example
(?(?=regex)then|else)	If the lookahead succeeds, the "then" part must match for the overall regex to match. If the lookahead fails, the "else" part must match for the overall regex to match. Not just positive lookahead, but all four lookarounds can be used. Note that the lookahead is zero-width, so the "then" and "else" parts need to match and consume the part of the text matched by the lookahead as well.	(?(?<=a)b|c)matches the second b and the first c inbabxcac
(?(1)then|else)	If the first capturing group took part in the match attempt thus far, the "then" part must match for the overall regex to match. If the first capturing group did not take part in the match, the "else" part must match for the overall regex to match.	(a)?(?(1)b|c)matches ab, the first c and the second c inbabxcac

Comments

Syntax	Description	Example
(?#comment)	Everything between (?# and) is ignored by the regex engine.	a(?#foobar)bmatches ab

Beispiele

Das nachfolgend aufgeführte Feld "Katalog" als Feldname ist nur in der Katalog-Filter-Funktion nutzbar. Die Nutzung von Expressions lässt sich hiermit aber perfekt zeigen.

Ziel	Beispiel	Erklärung
Automatische Markierung im Tai-Pan-Katalog aller Deutschland-Kataloge	Katalog~^04[1,4,6]{1}.(?!.*Stoxx.*)(?!.*Parkett.*)	Maskiert alle Märkte startend mit 04, gefolgt von 1, 4 oder 6. Dann mit einem beliebigen Zeichen und ausschließlich *STOXX* und *Parkett*, womit "044 DAX Xetra" markiert wird, "044 DAX Parkett" jedoch nicht 
Automatische Markierung im Tai-Pan-Katalog aller Leitdaten	Katalog~[0-9]{3}.(Indizes|Devisen|Rohstoffe|Futures.*|Renditen).[0-9]	Es werden alle Kataloge markiert die mit 3 Zahlen beginnen, einem beliebigen Zeichen fortgesetzt werden und dann mit Indizes oder Devisen fortgesetzt werden. Am Ende muss nochmals mindestens eine Zahl folgen nach einem beliebigen Zeichen (hier Leerzeichen). 

Ziel	Beispiel	Erklärung
Automatische Markierung aller relevanter Märkte	Katalog~^(04[1,4,6]{0,3}|[0-9]{3}.Österreich|[0-9]{3}.Dänemark|[0-9]{3}.Schweiz|[0-9]{3}.Indizes.[0-9]|[0-9]{3}.Devisen.[0-9]|[0-9]{3}.Rohstoffe.[0-9]|[0-9]{3}.Futures.[0-9]|[0-9]{3}.Zinsen|[0-9]{3}.Deutschland)(?!.*Parkett.*).*	Start immer mit 3 Ziffern und Ausschluß der Parkett-Kataloge für die 041,044,046-Kataloge. In den anderen sind .* Joker gesetzt und teilweise wird wie bei Indizes nach einem beliebigen Zeichen eine Ziffer verlangt 