online Übungssystem & online Prüfungssystem

# Tabellen filtern

**Syntax für Suchfelder in Tabellenspalten**

**Autor:**
Immo Schulz-Gerlach

**Version:**
1.0 – 08.08.2025

FernUniversität in Hagen

# Inhaltsverzeichnis

# Suchfelder zum Filtern von Tabellen nutzen

## Worum geht's?

Im Online-Übungs-/Prüfungssystem finden Sie in diversen Tabellenansichten im Betreuer- oder Korrektorenzugang Suchfelder zum Filtern von Tabellen anhand von Spalteninhalten. Das kann z.B. wie folgt aussehen:


*Tabellenkopf mit Suchzeile*

Hier können Sie Text bzw. Zahlen eingeben, um die Tabelle zu filtern, also alle Tabellenzeilen auszublenden, in denen der Suchbegriff nicht (in dieser Spalte) vorkommt. Aber diese Felder können noch mehr als nur reinen Suchtext entgegen zu nehmen und auf ein „Enthaltensein“ als Teilwort zu prüfen. Mit verschiedenen *Suchoperatoren* können Sie komplexere Suchen / Filteroperationen ausführen. Eine Auswahl praktischer Operatoren soll hier zusammengestellt werden.

## Hinweis: Zwei unterschiedliche Implementierungen

Es kann leichte Unterschiede im Such-/Filterverhalten je nach Tabellentyp geben:

- Bei vollständig in den Browser geladenen Tabellen (die den häufigsten Fall darstellen) erfolgt die Suche clientseitig in einem JavaScript.
- Es gibt aber auch Tabellen, die so umfangreich sind, dass zur Performance-Optimierung standardmäßig nur ein Tabellenausschnitt geladen und angezeigt wird und beim Blättern zwischen den „Seiten“/Ausschnitten der Tabelle diese immer einzeln vom Server nachgeladen werden. Diese sind natürlich nicht in ihrer Gesamtheit per JavaScript im Browser durchsuchbar, sondern hier wird die Suche an den Server delegiert.
  (Derartige Tabellen können Sie übrigens daran erkennen, dass darunter ein Link »Tabelle vollständig in Browser laden…« angeboten wird.)

Normalerweise braucht Sie dieser Unterschied nicht zu kümmern, aber da die Suchroutinen (serversitig vs. clientseitig) nicht dieselben sind, könnte es im Einzelfall Unterschiede im Verhalten oder in den unterstützten Operatoren geben. So unterstützt die JavaScript-Implementierung noch mehr Operatoren/Schreibweisen als die serverseitige Suche. Die im folgenden vorgestellten Operatoren/Schreibweisen sollten aber in beiden Varianten weitgehend gleich funktionieren.

# Ausgewählte Suchoperatoren

## Logische Operatoren

| Operator | Bedeutung | Beispiel |
|---|---|---|
| `\|` oder `OR` | Oder | `eier\|eyer` findet Tabellenzeilen, die in dieser Spalte die Teilworte `eier` oder `eyer` enthalten, z.B. Namen wie »Meier«, »Lohmeier«, »Meyer«, »Meyering« etc.. `<10 \| >20` findet in einer Ganzzahlen-Spalte Zeilen mit Werten kleiner als Zehn oder größer als 20. |
| `&&` oder `AND` | Und | `Rai && mund` findet Texte, die »Rai« *und* »mund« enthalten, aber nicht notwendig zusammenhängend, also nicht nur »Raimund«, sondern auch z.B. »Rainer aus Dortmund«. (Auch die Reihenfolge ist beliebig.) |
| `!` | Nicht | `!Hans` findet nur Tabellenzeilen, in denen das Teilwort »Hans« *nicht* vorkommt. `Rai && !mund` findet Tabellenzeilen, in denen zwar das Teilwort »Rai«, aber *nicht* zusätzlich »mund« vorkommt, d.h. Namen wie »Rainer« oder »Raimond« werden gefunden, »Raimund« dagegen nicht. |

Die logischen Operatoren sind hier nach aufsteigender Priorität sortiert, d.h. wenn Sie, wie z.B. im letzten Beispiel, die Operatoren `!` und `&&` kombinieren, bindet das Nicht stärker als das Und. (Vergleichbar mit »Punktrechnung vor Strichrechnung«, nur würde man hier eben sagen: »Nicht vor Und vor Oder«.)

## Vergleichsoperatoren / -Syntax

- Wenn Sie einfach nur Text eintippen, wird wie gesagt nach Tabellenzellen gesucht, die mindestens diesen Text enthalten, d.h. es werden auch **Teilübereinstimmungen** gefunden, genauer: Zellen, die den Suchbegriff als Teilstring enthalten.
  Neben dieser Teilstring-Prüfung gibt es auch noch komplexere Möglichkeiten, **Teilübereinstimmungen mit Suchmustern** zu finden:
  - **Wildcards** ermöglichen relativ einfache Suchmuster: Fügen Sie dazu ein `*` als Platzhalter für beliebig viele Zeichen oder ein `?` als Wildcard für genau ein beliebiges Zeichen in den Suchbegriff ein, z.B. `?atha*` passt auf Namen wie »Nathalie« oder »Katharina«.
    - Wenn Sie zur Vollübereinstimmungssuche (s.u.) den Suchbegriff in Anführungszeichen schreiben, werden diese Zeichen wörtlich als Teil des Suchbegriffs und nicht als Wildcard interpretiert.
    - Sollten Sie statt Anführungszeichen den `=`-Operator verwenden (s.u.), gibt es tatsächlich einen Unterschied zwischen serverseitiger und clientseitiger Suche (vgl. Hinweis: Zwei unterschiedliche Implementierungen): Die serverseitige Suche unterstützt Wildcards hinter `=`, die clientseitige JavaScript-Suche nicht – die behandelt `=` und `"` als äquivalent.
  - Komplexere Suchmuster können als sog. **Reguläre Ausdrücke** formuliert werden. Eine Beschreibung von regulären Ausdrücken wäre hier zu komplex, aber wer sich mit dieser Sprache auskennt, kann sie für Suchmuster wie folgt verwenden: Schreiben Sie im Suchfeld den regulären Ausdruck einfach zwischen zwei Querstriche.
    - Beachten Sie dabei, dass hier nach allen Tabellenzeilen gesucht wird, die in der durchsuchen Spalte einen Match des Musters als *Teilübereinstimmung* enthalten, z.B. `/[NK]atha/` sucht nach allen Vorkommen von z.B. »Katha« oder »Natha«

als Teilübereinstimmung, findet also z.B. »Katharina«, »Nathalie« oder »Anna-Katharina« etc.

- Suchflags können *hinter* dem schließenden Slash angegeben werden, z.B. `i` für „Ignore Case“, also Ignorieren der Groß/-Kleinschreibung. So findet `/[nk]atha/i` z.B. auch »Katharina« oder »nathAliA« etc.

- Um eine *Vollübereistimmung mit dem Suchmuster* zu verlangen, können Sie im regulären Ausdruck „Anchors“ verwenden: Ein `^` am Beginn des regulären Ausdrucks bedeutet z.B., dass das nachfolgende Muster den Beginn des Suchtreffers darstellen muss (kein Text vorangehen darf), ein `$` am Ende markiert entsprechend, dass ein Suchtreffer mit diesem Muster enden muss.
  Beispiel `/th$/` sucht nach allen Tabellenzeilen, die in der durchsuchen Spalte einen Text enthalten, der auf »th« endet. Oder: `/^[NK]atha\w+$/` sucht alle Tabellenzeilen, die in der durchsuchten Spalte einen Text enthalten, der mit »Natha« oder »Katha« beginnt, worauf weitere Wortzeichen (Buchstaben, Bindestrich etc., aber kein Leerzeichen und somit kein weiterer Name mehr) folgen dürfen. So werden z.B. »Katharina« und »Nathalie« gefunden, »Anna-Katharina« oder »Nathalie Sofie« hingegen werden ausgefiltert.

- Für reguläre Ausdrücke kann die Art der Tabelle wie oben im Hinweis zu zwei unterschiedlichen Implementierungen beschrieben eine Rolle spielen: Die serverseitige Suche in großen, nur seitenweise abgerufenen Tabellen verwendet den Java-, die browserseitige Suche den JavaScript-Flavour. Außerdem gilt für die serverseitige Suche derzeit, dass der gesamte Suchfeld-Inhalt ein regulärer Ausdruck in zwei Slashes sein muss, dass also nicht mehrere reguläre Ausdrücke mit z.B. logischen Operatoren kombiniert werden können.

- Wenn Sie einem Suchbegriff ein Gleichheitszeichen voranstellen (oder hinten anstellen) oder diesen Suchbegriff in Anführungszeichen einschließen, wird eine **Vollübereinstimmung** verlangt. Z.B. `=Anton` oder `Anton=` oder `"Anton"` sucht nur nach Zellen, in denen *genau/ausschließlich* der Text »Anton« steht, Zellen mit z.B. »Antonia« werden dann ausgefiltert.
  - Auch dies kann wieder mit einem Ausrufezeichen negiert werden. So sucht `!=Anton` oder `!"Anton"` wieder nach allen Zellen, die *nicht* genau das Wort »Anton« enthalten.
- Die Operatoren `>`, `<`, `>=` und `<=` können für Vergleiche herangezogen werden.
  - Bei Text bezieht sich das typischerweise auf eine alphabetische Ordnung. Der Suchbegriff `>Berthold` wird also nur auf Tabellenzellen zutreffen, deren Inhalt bei alphabetischer Sortierung hinter »Berthold« einsortiert würde, also Namen wie »Anton« oder »Berta« ausfiltern.
  - In Zahlenspalten sollte – sofern diese korrekt erkannt werden – tatsächlich ein numerischer und kein alphabetischer Vergleich erfolgen.
  - In Datumsspaten sollte – sofern diese korrekt erkannt werden – entsprechend eine zeitliche Ordnung zugrunde gelegt werden, also z.B. `>=15.05.2025` alle Zeilen mit einem Datum vor diesem 15. Mai ausfiltern.