Übersicht Reguläre Ausdrücke

Nachfolgend eine Übersicht der wichtigsten Befehle für Reguläre Ausdrücke.

Operator	Funktion	Beispiel
.	Wildcard - passt zu jedem Zeichen	/h.llo/ - Passt zu allen Texten die *h beliebiges Zeichen* gefolgt von *llo* enthalten
[ ]	Überprüft ob einer der Zeichen enthalten ist	/h[ae]llo/ -Passt zu *hallo* und *hello* /[A-Za-z0-9]/ - Erlaubt einen Großbuchstaben, Kleinbuchstaben oder Zahl
^	Überprüft den Anfang des Textes. Kann ebenfalls für nicht stehen.	/^test/ - Der Text muss mit *test* beginnen /hall[^aeiou]/ - Die Buchstaben *hall* dürfen nicht mit a, e, i, o oder u enden
$	Überprüft den Ende des Textes	/test$/ - Der Text muss mit *test* aufhören
|	Ermöglicht Alternativen	/(der|das)/ -Passt zu *der* und *das* /Kind(er|ergarten|le)/ - Passt zu *Kinder*, *Kindergarten* und *Kindle*.
?	Vorheriges Zeichen ist optional	/iPhone[1-7]?/ -Passt zu *iPhone*, *iPhone2* usw. bis *iPhone7*
*****	Wiederholung des vorherigen Elements (0 oder häufiger mal)	/Windows [0-9]*/ - Passt zu *Windows*, *Windows 98* und *Windows 7*, aber nicht Windows7.
+	Wiederholung des vorherigen Elements (1 oder häufiger mal)	/[0-9]+/ - Passt zu allen natürlichen Zahlen.
{n}	Exakt n-mal Wiederholung des vorherigen Elements	/[0-9]{3}/ - Passt zu allen 3 stelligen Zahlen.
{m,n}	Wiederholung des vorherigen Elements mindestens m-mal, maximal n-mal.	/[0-9]{1,4}/ - Passt zu allen 1 bis 4 stelligen Zahlen.

Überprüfung der Postleitzahl

Möchten wir überprüfen ob eine Eingabe eine (deutsche) Postleitzahl ist, so sähe ein möglicher Ausdruckt wie folgt aus: preg_match("/^[0-9]{5}$/", $eingabe);. Durch das ^ und das $ stellen wir sicher, dass nur die angegeben Zeichen vorkommen dürfen. Mit [0-9]{5} erlauben wir dann alle 5 stelligen Zahlen. Die 00000 würde hier leider auch erlaubt werden, obwohl es keine gültige PLZ ist.

Überprüfung der Telefonnummer

Eine Telefonnummer zu überprüfen ist schon etwas komplizierter, da diese am Anfang ein + für den Ländercode enthalten kann. Ebenfalls wird bei der Angabe dieser gerne auch Leerzeichen, Bindestriche oder Slashs verwendet. Ein möglicherer regulärer Ausdruck könnte so aussehen:

preg_match("/^\+?([0-9\/ -]+)$/", $eingabe);

Um hier das + Zeichen für die Ländervorwahl sowie den Slash innerhalb einer Rufnummer nutzen zu können müssen wir diese mittels \ vorab escapen. In diesem regulären Ausdruck beginnen wir also mit einem optionalen +, danach können dann Zahlen, Leerzeichen, Bindestriche und Slashs folgen.

Überprüfung der E-Mail-Adresse

Das Überprüfen auf eine gültige E-Mail-Adresse ist nicht trivial mittels regulären Ausdruck. Zum Glück aber gibt es in PHP die filter_var Funktion. Möchte ihr dennoch einen regulären Ausdruck für die E-Mail Adresse, so sähe der korrekte Code wie folgt aus:

```
1 preg_match("/^(?!(?:(?:\\x22?\\x5C[\\x00-\\x7E]\\x22?)
|(?:\\x22?[^\\x5C\\x22]\\x22?)){255,})(?!(?:(?:\\x22?\\x5C[\\x00-
\\x7E]\\x22?)|(?:\\x22?[^\\x5C\\x22]\\x22?)){65,}@)(?:(?:[\\x21\\x23-\\x27\\x2A\\x2B\\x2D\\x2F-
\\x39\\x3D\\x3F\\x5E-\\x7E]+)|(?:\\x22(?:[\\x01-\\x08\\x0B\\x0C\\x0E-\\x1F\\x21\\x23-\\x5B\\x5D-
\\x7F]|(?:\\x5C[\\x00-\\x7F]))*\\x22))(?:\\.(?:(?:[\\x21\\x23-\\x27\\x2A\\x2B\\x2D\\x2F-\\x39\\x3D\\x3F\\x5E-
\\x7E]+)|(?:\\x22(?:[\\x01-\\x08\\x0B\\x0C\\x0E-\\x1F\\x21\\x23-\\x5B\\x5D-\\x7F]|(?:\\x5C[\\x00-
\\x7F]))*\\x22)))*@(?:(?:(?!.*[^.]{64,})(?:(?:(?:xn--)?[a-z0-9]+(?:-+[a-z0-9]+)*\\.){1,126}){1,}(?:(?:[a-z][a-z0-
9]*)|(?:(?:xn--)[a-z0-9]+))(?:-+[a-z0-9]+)*)|(?:\\[(?:(?:IPv6:(?:(?:[a-f0-9]{1,4}(?::[a-f0-9]{1,4}){7})|(?:(?!(?:.*[a-f0-
9][:\\]]){7,})(?:[a-f0-9]{1,4}(?::[a-f0-9]{1,4}){0,5})?::(?:[a-f0-9]{1,4}(?::[a-f0-
9]{1,4}){0,5})?)))|(?:(?:IPv6:(?:(?:[a-f0-9]{1,4}(?::[a-f0-9]{1,4}){5}:)|(?:(?!(?:.*[a-f0-9]:){5,})(?:[a-f0-
9]{1,4}(?::[a-f0-9]{1,4}){0,3})?::(?:[a-f0-9]{1,4}(?::[a-f0-9]{1,4}){0,3}:)?)))?(?:(?:25[0-5])|(?:2[0-4][0-9])|(?:1[0-
9]{2})|(?:[1-9]?[0-9]))(?:\\.(?:(?:25[0-5])|(?:2[0-4][0-9])|(?:1[0-9]{2})|(?:[1-9]?[0-9]))){3}))\\]))$/iD';", $eingabe);
```

Dies ist in der Tat furchtbar kompliziert, eine vereinfachte Version wäre:

```
1 preg_match("/^[a-zA-Z0-9_.+-]+@[a-zA-Z0-9.-]+.[a-zA-Z]+$/", $eingabe);
```

Hier überprüfen wir zuerst, dass ein Teil vor dem @-Zeichen existiert, gefolgt von der möglichen Domain und der Domainendung. Umlaute werden bei dieser vereinfachten Variante leider nicht ermöglicht.