Tonis kleines Programmierhandbuch
- Ein Tutorial -

Inhalt

Tonis kleines Programmierhandbuch
- Ein Tutorial -

Vorweg: Um was gehts?

Diese Tutorial richtet sich an alle, die noch nie mit irgendeiner Programmiersprache gearbeitet haben und soll etwas die Angst vorm scripten in IPS (IP-Symcon) nehmen. Hier werden vor allem erste Schritte und die Grundlagen der Informatik erklährt. Untertitel des Tutorials könnte also lauten: Lerne zu denken wie ein Computer. PHP lernen wir hier eher spielend nebenbei. Es ist sinnvoll diesen Text nicht einfach blind zu lesen, sondern IPS zumindest schon einmal live gesehen und ein wenig darin herumgeklickt zu haben...

Was brauche ich zum scripten mit IPS?

Garnichtmal so viel. Ein Buch? Für den Einstig in IPS reicht die Doku und das Forum. Ein Einsteiger PHP-Buch schadet nicht, hilft vermutlich aber auch nicht so viel wie man meint - später aber sinnvoll. Es gibt viele tolle Programme, die Web-Entwickler benutzen um PHP zu programmieren. Brauchen wir nicht. IPS hat einen Scripteditor, der völlig ausreichend ist und obendrein IPS-Interne Funktionen unterstützt und einen Execute-Button, der unsere Scripte gleich testet. ausserdem bekommt man in einer Box mitgeteielt in welcher Zeile sich noch Fehler verstecken. Viel mehr können diese Tools auch nicht und irgendwelche integrierten Internet-Tools brauchen wir nicht. Was will man also mehr? Wir haben quasi ein Rundum-Sorglos-Paket schon an der Hand.

Mehr nicht? Aber dann ist das doch bestimmt alles schwierig, oder?

Moooooment...Zunächst müssen wir uns darüber klar werden, dass PHP eine wirklich mächtige und umfangreiche Sprache ist. Wir benötigen für IPS aber keine 5% davon um zum Beispiel das Licht, die Heizung, einige Unterhaltungselektronik und sogar die Gartenbewässerung zu bedienen. Und das alles in Abhängigkeit vom Wetter, von den Tageslichtverhältnissen, der Innen- und Außentemperatur, Bewegung oder persönlichen Einstellungen wie Tagschaltung, Nachtschaltung, Spätschicht, Wochenende, Urlaub, Anwesend, Abwesend, Kind kommt von der Schule und vieles mehr. Eine ganze Menge also fast ohne PHP... Aber eben nur fast.

Wer irgendwann fit ist mit PHP kann die restlichen 95% aber auch in IPS benutzen um mehr komfort einzubauen. So gibt es im IPS-Forum zum Beispiel eine Rubrik wo interessante Scripte veröffentlicht werden. Eines davon schaut zum Beispiel ob bei unserem E-Mail Anbieter neue Nachrichten vorhanden sind, ein Anderes kommuniziert mit der Telefonanlage und ein wiederum Anderes steuert eine Hifi-Anlage.

Ohne PHP? Wie denn dann?

Da müssen wir etwas ausholen. Alles was man mit IPS steuern kann, wird über zusätzliche, integrierte IPS-Funktionen bedient, die mit PHP rein garnichts zu tun haben. Diese Funktionen sind, wenn man so will, wie Hebel einer großen, auf den ersten Blick komplizierten, Dampfmaschiene. Wenn wir die IPS-Installation abgeschlossen haben und das Menü sich zu ersten mal öffnet fühlen wir uns ein bisschen so wie ein Durchschnittsmensch um 1900, der zum ersten mal so ein furchteinflößendes, zischendes Dampf-Monster sieht. Wir haben Respekt und sind vielleicht sogar ein wenig erschrocken. Wer sich die Maschiene nun genauer anschaut wird schnell feststellen, dass jeder Hebel eine bestimmte Aufgabe hat. Einige Hebel sind schon bis aufs letzte verschlissen weil er offenbar ständig bedient wurde, andere wiederum sehen fabrikneu aus.

Tonis kleines Programmierhandbuch
- Ein Tutorial -

Dieser kleine Exkurs soll nur verdeutlichen, dass auch der Computer auch nur eine Maschiene ist, die genau das tut was man ihm sagt - Was man sagt, nicht was man will! Um die Kommunikation zu vereinfachen gibt es eine Reihe von Funktionen die IPS für uns bereithält. Und jetzt darf man nicht denken dass eine Maschiene mit vielen Hebeln komplizierter ist als eine mit nur wenigen... Andersherum! Um so mehr Funktionen ich zur Auswahl habe desto eher ist doch eine dabei, die genau das tut was ich suche. Man muss nur wissen welche man nimmt. Es gehört zum "guten Ton" Funktionen möglichst treffend zu benennen. Das bedeutet man kann anhand des Namens schon erkennen was sie tut. Aber ohne Handbuch gehts trotzdem nicht. Dazu gibts die IPS-Doku online, in der jede einzelne Funktion erklährt ist. Und es ist nicht so, dass es immer nur einen richtigen Weg gibt. Es führen immer viele Wege zum Ziel. Und ob man in dem gleichen Script nun 20 Zeilen verwendet oder das die Aufgabe in nur einer Einzigen löst spielt keine Rolle. Man muss nur gewisse Regeln befolgen, die hier noch erläutert werden sollen. Nur Mut!

Aber was zum Geier ist eigentlich eine Funktion?

Eine Funktion ist eine Sammlung von einzelnen Befehlen die zusammengafasst eine definierte Aufgabe erledigen. So könnte, um auf unsere Dampfmaschiene zurückzukommen, der Vorgang Klappenverriegelung lösen, Klappe öffnen, Wasserkran heraziehen, Wasserventil öffen, warten bis voll, wasserkran zurückschieben, Klappe schließen, Klappe verriegeln als eine Funktion mit dem Namen "Wasserfassen" zusammengefasst werden. Und wenn man seinem Heizer sagen kann:"Los geh Wasserfassen" - dann interessieren uns die einzelnen Handgriffe eigendlich garnicht mehr und wir können uns wichtigeren Dingen zuwenden. Das macht uns die Sache doch einfacher, oder nicht?

Nun stellen wir uns aber mal vor wir haben einen übergenauen Heizer... Wir sagen "Wasserfassen" und er zuckt nur mit den Schultern: "Welches Wasser? Welche Dampfmaschiene?" - Auf soeinen Heizer kann man sich eben nicht blind verlassen. Wir müssen ihm also sagen welches Wasser und welche Dampfmaschiene und wir müssen wissen ob er verstanden hat was wir von ihm wollen indem wir von ihm sagen lassen ob alles geklappt hat. Einige Heizer haben also einen sogenannten Rückgabewert. Das kann ein einfaches Ja/Nein sein oder auch eine Meldung wie "Sir, 800 Liter, Sir!". Damit können wir arbeiten.

Ein Beispiel zum Anfassen:

PHP-Code:

```
$Liter = Wasserfassen("Wasserkran", 1);
```

Zitat:

> Geh Wasserfassen und hol das Wasser aus dem "wasserkran". Befülle damit die Dampfmaschiene Nummer 1 und melde mir wieviel Liter du eingefüllt hast. - STOP -

Tonis kleines Programmierhandbuch
- Ein Tutorial -

Da wäre ich ja nie drauf gekommen - woher weiss ich wie ich das schreiben muss?

Wir müssen die Syntax eingalten. Die Syntax ist die Grammatik der Programmiersprache. Wenn jemand deutsch spricht ohne sich auch nur grob an die Grammatik zu halten braucht man eine gewisse Phantasie um zu verstehen was der von uns will. Nun, der Computer hat hat ungefähr so viel Phantasie wie ein Blumentopf... Die Funktionen erkennen die übergebenen Informationen anhand ihrer Reihenfolge. Das bedeutet es ist nicht egal in welcher Reihenfolge wir ihnen die Informationen geben. Als erstes das "woher" danach das "wohin". Wie genau, das muss man vorher in Erfahrung bringen. Das Semikolon hinter der Anweisung sagt PHP, dass ihr nun fertig seid mit der formulierung der Anweisung. Da man zu besseren Übersicht - für uns, nicht für den Computer - immer nur eine Anweisung pro Zeile schreibt, kommt also an das Ende jeder Zeile ein Semikolon. Wie im Telegramm - Stop -

Sicherlich sind dir die Anführungszeichen aufgefallen, oder? Was hat es damit auf sich? Nun, laut Syntax müssen Texte, sogenannte Strings (zu deutsch: Zeichenketten), in Anführungszeichen gesetzt werden damit man erkennen kann wo ein String beginnt und wo er aufhört. Dazu gibt es ein paar Sonderregeln, auf die ich hier aber nicht näher eingehen werde. Die Nummer der Dampfmaschiene hingegen wird von der Funktion als Integer (Ganzzahl) erwartet. Da dürfen keine Anführungszeichen gesetzt werden. Denn "1" wäre ja kein Integer mehr sondern ein String! Auch die Eins ist ja ein Zeichen und könnte theoretisch verkettet werden, richtig?

Man kann auch Kommazahlen (man spricht vom Typ Float) benutzen. Aber eigentlich müsste man sagen Punktzahlen, denn in der Informatik wird 1.5 statt 1,5 geschrieben. Wir kennen das alle von den Versionsnummern von Software, gell? Aber nicht verwechseln! Kommazahlen sind keine Ganzzahlen, auch nicht wenn es eine 1.0 oder 1.000 ist - und schon garnicht wenn es eine "1.0" oder "1,0" ist. Naaa, verwirrt?

Damit sind wir auch schon beim nächsten Thema - den Variablen.

Eine Variable ist eine Art "Merker". Sie merkt sich Werte aller Typen und Längen. PHP übernimmt für uns das Management. Das bedeutet es wird Platz im Arbeitsspeicher angefordert und sich eine Adresse (wie eine Post-Adresse) gemerkt, damit man diese einmal gemerkten Werte auch irgendwann mal wiederfindet. Von alle dem bekommen wir aber nichts mit. Wir müssen uns nicht darum kümmern - Wie müssen sie nicht einmal anmelden (deklarieren) wie es bei anderen Programmiersprachen üblich ist. Das einzige was wir tun müssen ist der Variablen einen Namen zu geben. Wir haben sie $Liter genannt. Das Dollarzeichen macht PHP darauf aufmerksam, dass es sich bei "Liter" nicht um eine Funktion oder soetwas handelt sondern eben um eine Variable. Alle Variablen müssen durch ein $ gekennzeichnet sein!

Tonis kleines Programmierhandbuch
- Ein Tutorial -

Vorsicht Falle!

In IPS gibt es zwei Arten von variablen. Die eben beschriebenen Variablen sind so wie wir sie eben kennengelernt haben nur innerhalb unseres Scriptes bekannt. IPS hat aber noch eigene Variablen, die im Menüpunkt "Variables" verwaltet werden können. Diese Beiden Variablen-Arten haben nichts miteinander zu tun. Wenn man auf eine IPS-Variable zugreifen will geht das nur mit einem Dolmetscher. Genaugenommen gibts da mehrere Hochspezialisierte Dolmetscher, die wie unser Heizer von vorhin arbeiten, denn es sind auch Funktionen. Auch diese Dolmetscher-Funktionen sind IPS-Funktionen und somit eigentlich noch garkein PHP.

Eine dieser Funktionen heißt "getValueBoolean". Wie war das noch mit der Eindeutigkeit? Allein an ihrem Namen können wir schon sehen, dass sie etwas holt (symolisiert durch das "get") und zwar einen Wert (Value) - aber was ist ein Boolean? Bool-Werte, wir hatten das anfangs kurz angesprochen, speichern Ja/nein werte. Nur heissen sie nicht "Ja" und "Nein" sondern "true" (wahr) und "false" (falsch). Okay - die funktion holt sich einen Bool-Wert aus IPS. Und dann? Müssen wir ihr denn nicht sagen welche der unzähligen IPS-Variable wir haben wollen und müssen wir uns die Antwort unseres Dolmetschers nicht auch irgendwo merken? Ja, müssen wir!
PHP-Code:

```
$Regen = getValueBoolean("Regen");
```

Noch Fragen? Und nun ratet mal was "getValueString" oder "setValueBoolean" machen... Und immer noch kein PHP benutzt...

Waas? Wann gehts denn endlich mit PHP los?

Okay, aber erstmal ganz langsam, ja? Manchmal ist es sinnvoll erstmal zu wissen was in einer Variablen drin steht bevor man damit weiterarbeitet. Dafür kann man sich in IPS den Inhalt von Variablen in einem Output Window ausgeben lassen. Das ist besonders interessant wenn man einmal einen Fehler sucht. Denn dann sieht man warum sich ein script vielleicht einmal deneben benimmt.

PHP ist wie viele andere Programmiersprachen auch ansich eine Sammlung von unzähligen Funktionen. Eine dieser PHP Funktion ist "print". Print "druckt" den inhalt einer Variablen in unser Output Window.
PHP-Code:

```
$test = "Hallo Welt"; // ein Klassiker
print $test;
```

Zitat:

> merke dir einen String in einer Variablen mit dem namen "test". Gibt den Inhalt der Variablen $test im Output Window aus.

Nanu... was ist denn das mit dem Doppel-Schrägstrich (doppel slash - Sprich: släsch) da? Das ist ganz einfach. Manchmal, wenn zu Beispiel jemand Anderes auch verstehen soll was an einer Stelle passiert, ist es sinnvoll ein Script zu kommentieren. Alles was nach diesen Slashen kommt wird von PHP ignoriert. Ist, unter uns, auch ganz sinnvoll wenn man selbst nach einem Jahr mal wieder in das Script schaut. Denn dann hat man unter Umständen längst vergessen was das Script eigendlich macht, warum und vor allem wie. Wichtig beim Kommentieren ist vor allem, dass wir nicht nochmal in deutsch schreiben was dort passiert, denn das steht ja schon in PHP dort, sondern vielmehr **warum** das passiert.

Tonis kleines Programmierhandbuch
- Ein Tutorial -

Ein Kommentar soll ein Script erklären nicht übersetzen.

Was haben wir noch gemacht? Wir haben einer Variablen einen Wert zugewiesen. Gibts da noch etwas zu beachten? Höchstens, dass man immer der linken Seite zuweisen muss. Der Wert rechts vom "=" wird dem Wert links vom "=" zugewiesen - nicht andersherum!

Nun wird es Zeit mit dem Button oben rechts im Editor Bekanntschaft zu machen. Wenn wir da drauf klicken wird sich unten ein Fenster öffnen, wenn es nicht schon offen ist, und sich entweder über einen Fehler beklagen oder "Hallo Welt" ausgeben. Das war unsere erste PHP-Funktion.

Na Toll, das war ja aufregend... War das etwa schon alles?
Ohhh Nein, noch lange nicht! Eine Programmiersprache ist wie ein Werkzeugkoffer. Ein guter Handwerker hat für jede Aufgabe das richtige Werkzeug dabei. Aber eben auch einige ganz einfache Werkzeuge, ohne die es einfach nicht geht. Ein Hammer zum Beispiel ist alles andere als ein hochspezialisiertes Werkzeug. Es ist ein Allrounder, den man immer wieder mal für was brauchen kann. So einen Allrounder wollen wir uns nun einmal vornehmen. Dazu erstmal ein kleines Script, dass quasi schon Level Zwei in unserm kleinen Spiel ist.
PHP-Code:

```
$Regen = getValueBoolean("Regen");

if ($Regen == true)
{
  DoSomething();
}
```

Zitat:

> Hol eine Bool-Variable mit dem Namen "Regen" bei IPS ab und merke sie dir in $Regen. Wenn $Regen den Wert "true" hat, dann
> führe den Code zwischen den beiden klammern aus.

If, zu deutsch: Wenn. Das sagt doch schon alles, oder? Wenn also die Bedingung innerhalb der Klammern Wahr (true) ist, dann wird der Code zwischen den beiden geschweiften Klammern ausgeführt. Zu den Klammern gibt noch etwas zu sagen. Allerdings ist das eher Geschmackssache als eine Regel. Oftmals wird die erste geschweifte Klammer hinter die Bedingung geschrieben. Ich selbst schreibe sie immer in die Nächste Zeile und zwar aus zwei Gründen. Zu jeder geöffneten Klammer gehört immer eine geschlossene Klammer. Alles andere würde von PHP als Fehler angemäkelt werden. Wenn man die Klammern untereinander stehen hat sieht man sofort wo eine fehlt. Richtig nützlich wird diese Angewohnheit erst wenn man irgendwann mehrer Klammernblöcke ineinander verschachtelt. Wenn man dann weiterhin den Code zwischen den Klammern um zwei Leerschritte einrückt ist die Übersichtlichkeit nahezu perfekt. Das hiltf uns um uns in unserem Script zurechtzufinden und es erleichtert Anderen das lesen wenn ihr im Forum um Hilfe bittet. Einrücken ist suuuuper wichtig.

Tonis kleines Programmierhandbuch
- Ein Tutorial -

Exkurs:
Zitat:

> *Wo wir grad dabei sind. Wenn ihr euch mal komplett in eurem Script verstrickt habt und nicht weiter wisst, dann ist es sinnvoll den Code ins Forum zu posten und dabei ein Paar einfache Regeln zu beachten.*
>
> *Setzt den Code in PHP-Tags. Dann wird er von der Forensoftware so formatiert, dass man ihn leicht lesen kann. Die Tags könnt ihr bekommen wenn ihr im Forum oben rechts das kleine PHP-Symbol klickt und euer script zwischen den (php) Tag und den (/php) End-Tag kopiert.*
>
> *Ausserdem schreibt bitte dabei was ihr mit dem Script eigentlich erreichen wollt und wo ihr nicht weiterkommt. Vielleicht auch ein paar Worte dazu was ihr schon versucht habt. Dass man nicht nur schreibt: "Oha man alles Kacke, das geht nicht!" gebietet der Menschenverstand. Wenn ihr die Leser freundlich motiviert euch zu helfen habt ihr gute Chancen, dass euch schnell geholfen wird und vielleicht sogar ein Tip gegeben wird wie man es komplett besser machen könnte und wo euer Script vielleicht noch Schwächen hat. Das gilt übrigens für alle Foren im Internet.*

Was haben wir da mit dem "==" gemacht? Tippfehler? Nö... Damit PHP weiss dass wir nicht die Variable in den runden Klammern auf true setzen wollen sondern etwas miteinander zu vergleichen versuchen muss sich das "=" von dem "==" unterscheiden. Eine Sache, die man immer oft liest und, wenn man es nicht weiss, nicht versteht, will ich noch dazu sagen. Der Wert innerhalb der runden Klammer muss true sein, damit die Bedingung erfüllt ist. Wenn wir wie in diesem Beispiel eine Variable haben, die einen Bool-Wert beinhaltet, so wird verglichen ob der Wert innerhalb der Variablen true ist. Also (true == true). In diesem Fall kann man das "== true" und die Klammern weglassen. Es wird dann direkt der Wert in der variablen auf true oder false überprüft. Achja, die Geschichte mit dem Semikolon entfällt in der Zeile mit dem if.

Was wenn die Bedingung nicht erfüllt wird

Machmal ist es sinnvoll darauf zu reagieren wenn eine Bedingung nicht erfüllt wurde.
PHP-Code:

```
if $FensterOffen  // kurze Schreibweise (nur bei Boolean möglich)
{
  // Fenster schließen
}
else // nicht offen, also geschlossen
{
  // Fenster öffnen
}
```

Zitat:

> Wenn das Fenster offen ist führe den Code im oberen Klammernblock aus. Wenn es jedoch geschlossen ist, dann springe in den else-Teil und führe den Code in diesem klammernblock aus.

Klar, oder?

Tonis kleines Programmierhandbuch
- Ein Tutorial -

Boolsche Algebra

Keine Angst... So weit wollen wir nicht in die Materie einsteigen. Aber dass es soetwas gibt und wie man es auf unterstem Level benutzt wollen wir noch schnell behandeln. Es kommt häufig vor, dass man mehrer Faktoren hat, auf die man reagieren muss. Um diese möglichst übersichtlich (Übersicht ist Alles! Wirklich!) miteinander zu verknüpfen gibt es einige sogenannte "Operatoren", die wir auch aus dem täglichen Leben kennen.

PHP-Code:

```
if (($Regen == true) && ($FensterOffen == true))
{
  // mach das hier das Fenster zu
}
```

Zitat:

> Wenn die Variable $Regen den Wert true hat UND (&&) die Variable $FensterOffen den Wert true hat, dann führe den Code zwischen den Klammern aus.

Wichtig hierbei ist, dass die Bedingungen innerhalb der inneren klammern erfüllt werden und die Ergebnisse dieser Klammern zusammen wieder eine Bedingung, nämlich die in den Äußeren Klammern, ergeben, die ebenfalls erfüllt sein muss. Also wenn es regnet UND das Fenster offen ist, wird das Fenster geschlossen Weder wenn es regnet und das Fenster schon geschlossen ist, noch wenn das Fenster zwar offen ist, es jedoch nicht regnet, wird das Fenster geschlossen. Logisch, oder?

Ausser dem == (ist gleich?) und dem && (UND) Operator gibt es noch weitere. Die wichtigsten sind != (ungleich) und || (ODER). Für "&&" gibt es noch die schreibweise "and" und für "||" gibts noch "or". Was wir benutzen hängt vom persönlichen Geschmack ab.

Noch so ein Werkzeug: Die For-Schleife

Eine Schleife, in diesem Fall eine For-Schleife, macht nichts weiter als ein und den selben Code mehrmals auszuführen. Dafür muss das Script wissen wo wir anfangen wollen, wieweit wir zählen wollen und in welchen Schritten wir zählen.
Allereinfachstes Beispiel: Wir wollen einen Befehl genau 5 mal aufrufen. wir zählen also von 0 bis 5 (ohne die 5 selbst) in Einerschritten. Damit wir in der Schleife wissen wo wir gerade sind merken wir uns den Wert in einer variablen - exemplarisch "i". der Befehl, der ausgefürt werden soll wird in geschweifte Klammern geschrieben. Es können nämlich auch mehrer Befehle in diesem Befehlsblock stehen und das Script muss ja wissen wo es anfangen soll und wieweit es gehen soll.

PHP-Code:

```
for ($i=0; $i<5; $i++)
{
  // erstmal
  MachWasSinnvolles();
  // dann
  NochEtwasAnderes();
}
```

Tonis kleines Programmierhandbuch
- Ein Tutorial -

Zitat:

> Zähle in der Variablen i von 0 ($i=0) immer einen hinzu ($i++). Und das solange wie i kleiner als 5 ist ($i<5). Führe für jeden Durchlauf den Code aus, der zwischen { und } steht.

Aufpassen: 5 ist **nicht** kleiner als 5! Also zählt er nur von 0 bis 4 was in etwa genau so weit ist wie wenn er von 1 bis 5 zählen würde. Das könnte man eine Eigenart der Informatik nennen, weil dort fängt man immer mit der Null an.

Das ist nur ein Auszug aus dem ganzen Thema. Ich hab das "mal eben" zusammengetippt, grad wie es mir in den Sinn kam. Immer wenn mir was sinnvolles einfällt werde ich das hier ergänzen bzw korrigierern. Hinweise, Vorschläge oder Anregungen aber auch Danksagungen nehme ich gern per PM entgegen.

Da ich aber hier nicht ALLES abhandeln kann gibt es jetzt noch die:

Links und weiterführende Literatur :

Das offizielle Referenzmanual zu PHP: www.php.net/manual/de/
Ganz heisser Tip: www.selfphp.de
Auch erwähnenswert: www.php-dummies.de
Ein Tip von jheinz: http://tut.php-q.net/

"PHP 5.1 Kompendium"
Nachschlagewerk (also kein Tutorial) für Fortgeschrittene: bei Amazon
"PHP und MySQL für Kids"
hab ich selbst empfohlen bekommen - angeblich sehr simpel erklärt: bei Amazon
"PHP 5 leicht & verständlich"
Top Preis/Leistung - Die ganze Serie ist aus meiner eigenen Erfahrung sehr zu empfehlen: bei Amazon