PHP プログラミング

基礎

#2

9．ネットワーク基礎

Web に関わる上で、絶対に避けて通れない概念の一つとして URI(URL) があります .URI(URL) を理解するために最低現必要なネットワークの仕組みについて解説します .

URI(URL) とは

URI、URL とは、以下の用語の略称です .
URI…Uniform Resource Identifier
(リソースを一意に特定する形式)
URL…Uniform Resource Locator
(リソースの場所を示す形式)

上記の名称の通り、ネットワーク上でのリソースを指し示すための文字列です .URI のほうが広い概念で、URI の一部に URL は含まれます .URI の仕様は多岐にわたるので、ここでは URL に限定して説明します .

URL の基本形式

URL は
スキーム : スキーム別表現形式
という形で記述します .

ｈｔｔｐ：／／ｅｘａｍｐｌｅ．ｃｏｍ
を例に取ると、http がスキーム、//exapmle.com がスキーム別表現形式になります .

スキームとは、URL が指し示すリソースの種類を表します .URL では一般にリソースにアクセスするために使用されるプロトコルを記述します .Web ブラウザでアドレスバーに URL を入力する場合に、スキームを省略した場合には HTTP が自動で補完されます .

HTTP と HTTPS

ＨＴＴＰは Hyper Text Transport Protocol の略で、ハイパーテキストと呼ばれる相互にリンクした文書を転送するための方法として開発されました . その後、画像や動画なども転送できるなど、機能が拡張され現在に至ります .

ＨＴＴＰは通信時に内容をそのまま転送するため、個人情報やログイン情報を送ると盗聴される恐れがあります .HTTPS は通信時に通信内容を暗号化して盗聴を防ぐことができます . また、電子証明書と認証局という仕組みを使用して通信の相手先が正しいかをチェックすることができます . ただし、通信相手の行為が正しいことを証明しないことに注意してください .

http/https のスキーム別表現形式は、
// ユーザ名 : パスワード @ ホスト名 : ポート番号 / パス
となっていますが、ユーザ名、パスワード、ポート番号は特殊な場合を除いて省略されます .

ホスト名

URL では、どのコンピュータがそのリソースを保持しているかを表すために、ホスト名と言うものを使用します . ホスト名とは、DNS などによって解決される名前や、IP アドレスのことを指します .

IP アドレス

ネットワークにつながるコンピュータ (ホスト / ネットワークデバイス) には一台毎に最低一つの IP アドレスを持ちます . 現在主に使われている規格は IPv4 と言う規格で、最大で 42 億程度のホストを識別できます . しかしネットワークデバイスの普及に伴ないアドレスの枯渇という問題が発生しつつあるため、IPv4 の 2 の 96 乗倍 (約 340 澗) のデバイスを識別できる規格に移行しようとしています .

DNS(Domain Name System)

各ホストに割り当てられた IP アドレスには特に関連性がなく、ユーザが利用する上で不便です . そのた

め、ホストを表わす文字列と IP アドレスを紐付ける仕組み (名前解決) が考案されました . 古くは名前と IP アドレスを列挙したテキストファイルを共有することで管理していました . しかしホストの増大に伴ない管理が煩雑化したため、ネットワーク上に名前解決をする専用の機能をするサーバを設置し、分散管理する仕組みが開発されました . この仕組みが DNS です .

localhost

IP アドレスには特殊な用途のために使用されるアドレスがあります .127.0.0.1 と言う IP アドレスはそのデバイス自身を表す IP アドレスで、どのコンピュータで使用してもそのコンピュータ自身を指し示します .127.0.0.1 に対応するホスト名が localhost です .localhost もそのコンピュータ自身を表わします .

ポート番号

1 台のコンピュータで複数の通信を行なうためにポート番号というものを使用します . ポート番号は 0 ～ 65535 まで設定でき、一部の代表的なプロトコルではポート番号が予約されています (HTTP は 80 など).

前述したスキームと対応して自動的に決定されることが多いため、省略されても問題ありません . 運用上の都合により予約されているポート番号以外の番号が使われることもありますので、その場合には注意が必要です .

パス

基本的にはホスト上のドキュメントルートからのファイルパスを表します . ただし、最近では特殊な運用方法も一般化してきているため注意が必要です (REST など).

一般的にはパスは以下の構成を取ります .

ドキュメントルート直下のディレクトリ名またはファイル名 (0 個以上で "/" で区切られる)? クエリ文字列 (0 個以上で & で区切られる)

ディレクトリ名またはファイル名が "/" で終了している場合は注意が必要で、ファイル名が指定されていない場合はデフォルトドキュメントと呼ばれるファイルが自動的に選択されます . デフォルトドキュメントとは Web サーバで設定されているもので、index.php や index.html といったファイル名を使用することが一般的です .

クエリ文字列

クエリ文字列はファイルパスの後ろに "?" と共に記述される文字列で、変数名 1= 値 1& 変数名 2= 値 2 のような形式で記述されます .PHP では $_GET 変数を使うことで、このクエリ文字列で渡される値を取得できます .

日本語などのマルチバイト文字列は直接クエリ文字列として記述できないので、URL エンコード / デコードと言う変換を行ない URL で使用できる文字にします .

１０. データと制御構造１

コンピュータと数

コンピュータは情報を全て数値として扱います . いろいろな数は勿論 , 文章や音楽 , 写真なども沢山の数のあつまりとして処理されます .

プログラムを作成するうえで実際に文章や音楽などを直接数値として扱うことは稀ですが , 数値として保存されていることによる様々な問題や利点を知ることは重要です .

コンピュータで扱われる数は , 私たちが普段使用している数とは違うルールで表現されます .

私たちが普段使用しているのは , 1,2,3,4,5,6,7,8,9,10,11,12,13…といった形の数値表現です .10 になったときに一つ桁があがる 10 進法と言う表記方法です . 時間や月は 12 進法と言えますが ,10,11 が二桁になるので実際には 10 進表記による 12 進数と言えるでしょうか . 正確に 12 進表記にする場合には 10 を A,11 を B で置き換えるなどの工夫をします . 同様に分 , 秒は 60 進数になります .

コンピュータの場合は , 数字の種類がぐっと減って 2 進数で表現されます .0,1 の二種類の数字しか使いません . なぜこの二種類だけしか使わないかと言うと , 簡単に言ってしまえば「区別が付けやすいから」と言うことになります . コンピュータは電気で動いていますが , 電圧の高い低いを 0,1 に割り当てて計算します . もしこれが 10 進数だと , 電圧がどのくらい高いか , を 10 段階に割り振る必要があります . 割り振ったとおりに動いてくれれば良いのですが , 様々な原因で電圧はふらふらと変動するため , 別の数値になってしまうこともありえます . そのため , 両極端のどちら側か , という判断をします .

他にも , ハードディスクは磁力で情報を保存しますが ,N 極 ,S 極を 0,1 に割り当てていますし ,CD や DVD は穴のあるなしを 0,1 に割り当てています . このように ,2 進数は単純であるだけにとても便利に使用することができます .

しかし ,2 進数は 10 進数に比べて桁数が多くなってしまったり , ぱっと見で数の大きさかが分かり難いことから , 人間向けには 16 進数として表示することが一般的です .16 進数では 0 から 15 迄の数が一桁で表現できます . また ,10 から 15 までを一桁で表すために A から F に置き換えられます . なぜ 16 進数を使用するかという理由ですが ,16 進数の 1 桁がちょうど 2 進数 4 桁と同じ数を表現できるからです . 図 25 の 10 進数で 15 と 16 の行を見ると ,2 進数では 4 桁から 5 桁へ ,,16 進数では 1 桁から 2 桁へ揃って桁あがりしていることが判ると思います .

現在のコンピュータの仕組みとして ,2 進数 8 桁を一まとまりとして扱うことが基本となっています .2 進数一桁のことを 1 ビット (bit),8 桁のことを 1 バイト (byte) と呼んでいます (過去には 1 バイトが 8 ビットでは無いコンピュータもありました).

CPU や OS の説明に書かれている 32bit や 64bit は一度に扱える数の量を表していますし , ハードディスクやメモリの TB(テラバイト),GB(ギガバイト),MB(メガバイト) などは保存できるデータの量を表しています .

コンピュータと文字

コンピュータでは文章も数値で表していると前の項で説明しました . 具体的にどのように表しているのか , 図 25 の ASCII と言う列を見てください . 大文字の A は 10 進数で 65 と割り振られています . 小文字の a は 97 に割り振られています . つまり , コンピュータの中では ,A と a は別の番号が割り振られていて , 同じ A の大文字 / 小文字であると判断するためには工夫が必要になると言うことが判ると思います .

"ABC" と言う文章 (文字列と言います) は , 実際には 65,66,67 と言う数値の列としてコンピュータは処理をします . スペースは 32, 改行は 13 など , いくつかの特殊な文字 (制御文字) もコードが割り振られています .

英数字や一部の記号だけなら 7bit(0~127) で収まるのですが , 漢字やひらがな , 特殊な記号は 8bit でも足りません . そのため , 多バイト文字コードと言う 1 バイト以上を使って 1 文字を表わす方法が考え出されました .UTF-8 も多バイト文字コードの一つです .

図 25. 2 進数 ,10 進数 ,16 進数 ,ASCII コード

2進	10進	16進	ASCII	2進	10進	16進	ASCII	2進	10進	16進	ASCII	2進	10進	16進	ASCII
00000000	0	0	**NUL**	01000000	64	40	@	10000000	128	80		11000000	192	C0	
00000001	1	1	**SOH**	01000001	65	41	A	10000001	129	81		11000001	193	C1	
00000010	2	2	**STX**	01000010	66	42	B	10000010	130	82		11000010	194	C2	
00000011	3	3	**ETX**	01000011	67	43	C	10000011	131	83		11000011	195	C3	
00000100	4	4	**EOT**	01000100	68	44	D	10000100	132	84		11000100	196	C4	
00000101	5	5	**ENQ**	01000101	69	45	E	10000101	133	85		11000101	197	C5	
00000110	6	6	**ACK**	01000110	70	46	F	10000110	134	86		11000110	198	C6	
00000111	7	7	**BEL**	01000111	71	47	G	10000111	135	87		11000111	199	C7	
00001000	8	8	**BS**	01001000	72	48	H	10001000	136	88		11001000	200	C8	
00001001	9	9	**HT**	01001001	73	49	I	10001001	137	89		11001001	201	C9	
00001010	10	A	**NL**	01001010	74	4A	J	10001010	138	8A		11001010	202	CA	
00001011	11	B	**VT**	01001011	75	4B	K	10001011	139	8B		11001011	203	CB	
00001100	12	C	**NP**	01001100	76	4C	L	10001100	140	8C		11001100	204	CC	
00001101	13	D	**CR**	01001101	77	4D	M	10001101	141	8D		11001101	205	CD	
00001110	14	E	**SO**	01001110	78	4E	N	10001110	142	8E		11001110	206	CE	
00001111	15	F	**SI**	01001111	79	4F	O	10001111	143	8F		11001111	207	CF	
00010000	16	10	**DLE**	01010000	80	50	P	10010000	144	90		11010000	208	D0	
00010001	17	11	**DC1**	01010001	81	51	Q	10010001	145	91		11010001	209	D1	
00010010	18	12	**DC2**	01010010	82	52	R	10010010	146	92		11010010	210	D2	
00010011	19	13	**DC3**	01010011	83	53	S	10010011	147	93		11010011	211	D3	
00010100	20	14	**DC4**	01010100	84	54	T	10010100	148	94		11010100	212	D4	
00010101	21	15	**NAK**	01010101	85	55	U	10010101	149	95		11010101	213	D5	
00010110	22	16	**SYN**	01010110	86	56	V	10010110	150	96		11010110	214	D6	
00010111	23	17	**ETB**	01010111	87	57	W	10010111	151	97		11010111	215	D7	
00011000	24	18	**CAN**	01011000	88	58	X	10011000	152	98		11011000	216	D8	
00011001	25	19	**EM**	01011001	89	59	Y	10011001	153	99		11011001	217	D9	
00011010	26	1A	**SUB**	01011010	90	5A	Z	10011010	154	9A		11011010	218	DA	
00011011	27	1B	**ESC**	01011011	91	5B	[	10011011	155	9B		11011011	219	DB	
00011100	28	1C	**FS**	01011100	92	5C	\	10011100	156	9C		11011100	220	DC	
00011101	29	1D	**GS**	01011101	93	5D	]	10011101	157	9D		11011101	221	DD	
00011110	30	1E	**RS**	01011110	94	5E	^	10011110	158	9E		11011110	222	DE	
00011111	31	1F	**US**	01011111	95	5F	_	10011111	159	9F		11011111	223	DF	
00100000	32	20	**SP**	01100000	96	60	`	10100000	160	A0		11100000	224	E0	
00100001	33	21	!	01100001	97	61	a	10100001	161	A1		11100001	225	E1	
00100010	34	22	"	01100010	98	62	b	10100010	162	A2		11100010	226	E2	
00100011	35	23	#	01100011	99	63	c	10100011	163	A3		11100011	227	E3	
00100100	36	24	$	01100100	100	64	d	10100100	164	A4		11100100	228	E4	
00100101	37	25	%	01100101	101	65	e	10100101	165	A5		11100101	229	E5	
00100110	38	26	&	01100110	102	66	f	10100110	166	A6		11100110	230	E6	
00100111	39	27	'	01100111	103	67	g	10100111	167	A7		11100111	231	E7	
00101000	40	28	(	01101000	104	68	h	10101000	168	A8		11101000	232	E8	
00101001	41	29	)	01101001	105	69	i	10101001	169	A9		11101001	233	E9	
00101010	42	2A	*	01101010	106	6A	j	10101010	170	AA		11101010	234	EA	
00101011	43	2B	+	01101011	107	6B	k	10101011	171	AB		11101011	235	EB	
00101100	44	2C	,	01101100	108	6C	l	10101100	172	AC		11101100	236	EC	
00101101	45	2D	-	01101101	109	6D	m	10101101	173	AD		11101101	237	ED	
00101110	46	2E	.	01101110	110	6E	n	10101110	174	AE		11101110	238	EE	
00101111	47	2F	/	01101111	111	6F	o	10101111	175	AF		11101111	239	EF	
00110000	48	30	0	01110000	112	70	p	10110000	176	B0		11110000	240	F0	
00110001	49	31	1	01110001	113	71	q	10110001	177	B1		11110001	241	F1	
00110010	50	32	2	01110010	114	72	r	10110010	178	B2		11110010	242	F2	
00110011	51	33	3	01110011	115	73	s	10110011	179	B3		11110011	243	F3	
00110100	52	34	4	01110100	116	74	t	10110100	180	B4		11110100	244	F4	
00110101	53	35	5	01110101	117	75	u	10110101	181	B5		11110101	245	F5	
00110110	54	36	6	01110110	118	76	v	10110110	182	B6		11110110	246	F6	
00110111	55	37	7	01110111	119	77	w	10110111	183	B7		11110111	247	F7	
00111000	56	38	8	01111000	120	78	x	10111000	184	B8		11111000	248	F8	
00111001	57	39	9	01111001	121	79	y	10111001	185	B9		11111001	249	F9	
00111010	58	3A	:	01111010	122	7A	z	10111010	186	BA		11111010	250	FA	
00111011	59	3B	;	01111011	123	7B	{	10111011	187	BB		11111011	255	FB	
00111100	60	3C	<	01111100	124	7C	|	10111100	188	BC		11111100	256	FC	
00111101	61	3D	=	01111101	125	7D	}	10111101	189	BD		11111101	257	FD	
00111110	62	3E	>	01111110	126	7E	~	10111110	190	BE		11111110	258	FE	
00111111	63	3F	?	01111111	127	7F	**DEL**	10111111	191	BF		11111111	255	FF	

データ型

プログラムはデータを処理して結果を出すことが主な目的になります . データは入力されたもの , プログラムを作成するときに既に用意されたものなど様々です .

データの種類 (種類のことを型と言います) はひとつだけではありません . 最初の例で扱ったメッセージは文字列型のデータです . 文字列型の他には , 数値を表わす整数型や実数 (浮動小数点) 型 , ひとまとまりのデータのグループを表す配列などがあります .

また , 数値や文字列と並んで重要な型として , 論理型があります . 論理型はある事柄が正しいか正しくないかを表すために使用され , 論理型は TRUE(真),FALSE(偽) の二つの値しかありません . 論理型は後述する論理演算子と条件分岐の項目で詳しく説明します .

上記の他にも特殊な型として , データが無いことを表す NULL 型やデータと処理をまとめたオブジェクト型というものがあります . (表 1)

var_dump 関数を使うと、データの型を表示することができます (図 32, 図 26). var_dump 関数では、後述する変数の内容についての情報も表示することが出来るため、プログラムの開発中に動作を確認するために多用します . ただし、あくまでも動作確認のための関数ですので、プログラムが完成した時点でコメントアウトや削除をしておくようにして下さい .

表 1　よく使われるデータ型

名称		例
論理値	bool	TRUE , FALSE
整数	int	-1 , 0 , 1 , 2 , 3456789
実数	float	3.1414 , 1.4142
文字列	string	" 文字列 " , ' 文字列 '
配列	array	array('a','b','c')
オブジェクト	object	new MyClass()
ヌル	NULL	NULL

変数

データは文章や数値そのものを表わしますが , そのデータが何を表しているかはわかりません . 例えば ,100 という整数型のデータがあったとして , それが値段なのか量なのかは判別できません .

そこで , データに名前を付けることにします . 変数とはデータに名前を付ける機能です .

PHP では変数は "$" 記号のあとに ,"_"(アンダースコア) または英数字を組み合せて名前を付けます . "$" の後の一文字目はアンダースコアか英字のみ使用できます .

使用できる変数名
$_1
$price_1
$sumAllItems

使用できない変数名
$1　($ のすぐ後が数字)
$ 身長 (英数字以外を使用している)
$a c　(スペースを含んでいる)
$A+C　(アンダースコア以外の記号を含んでいる)

変数名をどのように付けるかは (ルールに則っている限りは) 自由ですが , 出来るだけ何のためのデータなのかわかりやすい名前を付けるようにしてくださ

図２６　program06.php の実行結果

```
論理値　bool(true) / bool(false)
整数　　int(123)
実数　　float(45.6)
文字列　string(9) " 文字列 "
配列　　array(3) { [0]=> string(1) "a" [1]=> int(123) [2]=> bool(true) }
オブジェクト object(DateTime)#1 (3) { ["date"]=> string(19) "2012-09-20 19:48:26" ["timezone_type"]=> int(3)
["timezone"]=> string(13) "Europe/Berlin" }
NULL　　NULL
```

い .$a や $b よりも $price や $tax などの方が良い変数名と言えます .

プログラム言語によっては , 変数自体に型があり , 代入できるデータ型が制限されることもありますが ,PHP では変数には型がありません . 数値型のデータが代入されている変数に文字列型の変数を代入することもできます .

入力の仕方のプログラムで使用した $_POST も実は特別な変数でした .

PHP では POST リクエストに含まれるデータ (パラメータ) は自動的に $_POST にセットされます .(GET リクエストでは $_GET 変数にセットされます)

ＰＨＰでは慣習として $_ から始まる変数名は特別なものとされていますので , 自分で作る変数には基本的には使用しないようにしてください .

データに名前を付けるためには , 変数にデータを代入します .

$price = 100;

$gram = 100;

このように , 変数名 = データ (変数や後述する式などでも可); で値を代入できます .

"="(等号) を使っていますが , 数学と異なり両端が等しいと言う意味ではないことに注意してください .(等しいことを表す記号は後述します)

また、代入は値をコピーすることに注意して下さい . $a = $b; のように、変数 $b の内容を $a に代入しても、$b の内容は無くなりません .

課題 1

目的："PHP プログラミング " と言う文字列を変数に代入した上で表示する

出来る事：

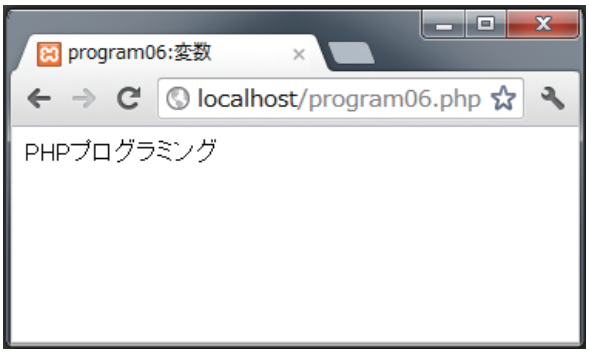
$ 変数名 = ' 文字列 ' ；で変数に代入できる .

echo $ 変数名 ；で変数を表示できる .

エディタで新規作成して ,xampp のドキュメントルートに "program07.php" というファイル名を付けて保存して下さい .(文字コードは UTF-8：BOM なしです)

図 33 を参考にプログラムを入力して上書き保存し , ブラウザで program07.php にアクセスしてみて下さい (図 27).

図 27　Ｐｒｏｇｒａｍ０７の実行結果



変数はそれだけで見ると有用性がわかりにくいですが、この後で学ぶ制御構造などと組みあわせることで、プログラムを記述するうえで必須とも言える機能といえることがわかると思います .

演算子

数値データを計算したり文字列を結合したりするには演算子と呼ばれる記号を使います .

四則演算に使う演算子は + , - , * , / が加減乗除に対応します . また割り算の余り (剰余) を求める場合は % を使用します .

計算の優先順位は数学と同様に乗除が先で加減が後になります . 計算の優先順位を変える (かけ算よりも足し算を先にするなど) には , これもまた数学と同様に括弧を使用します .

課題 2

目的：

以下の四則演算 , 剰余の計算結果を表示する

5 + 2

5 - 2

5 / 2

5 / 2 の余り

1 + 2 * 3

(1 + 2) * 3

出来る事：

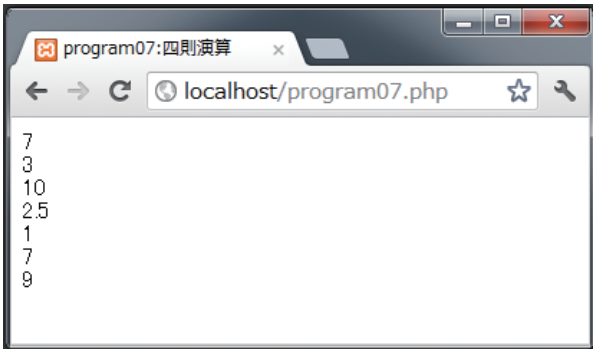
四則演算 + , - , * , /

剰余 %

エディタで新規作成して ,xampp のドキュメントルートに "program08.php" というファイル名を付けて保存して下さい .(文字コードは UTF-8：BOM なしです)

図 34 を参考にプログラムを入力して上書き保存し , ブラウザで program08.php にアクセスしてみて下さい (図 28).

図 28 Program08.php の実行結果



文字列の結合

すでに何度も使ってきましたが , 記号 "." は文字列の結合演算子と呼ばれ , 演算子の左右の文字列またはデータを文字列として結合します . 文字列の場合はそのまま結合し , 文字列以外のデータの場合は文字列になるようにデータを変換してから結合します .

課題 3

目的：

課題 2 に加えて , 計算式も表示する .

出来る事：

文字列の結合は "." を使用する

計算の順序を変えるには括弧を使う

【不具合解決コース】

エディタで新規作成して ,xampp のドキュメントルートに "program09a.php" というファイル名を付けて保存して下さい .(文字コードは UTF-8：BOM なしです)

図 35 を参考にプログラムを入力して上書き保存し , ブラウザで program09a.php にアクセスしてみて下さい (図 29).

図 29 Program09a.php の実行結果

なぜか計算式が表示されないケースと表示されるケースが出てきたと思います .

これは文字列の結合演算子の優先順位が加減と同じで , 同じ優先順位では左側の演算が先に行なわれるために起こります . つまり加減の計算よりも先に文字列と左辺の数字が結合され , その後で右辺と計算されます .

(http://www.php.net/manual/ja/language.operators.precedence.php)

乗除 , 剰余の計算では , 文字列の結合演算子より優先順いが高いため , 計算が先に行なわれて問題が発生しません .

括弧で計算順序を表わすと以下のようになります .

("5 + 2 = " . 5) + 2

また , 少々ややこしいのですが ,PHP のルールとして , 文字列と数値を計算した場合 , 文字列型は数字から始まる場合は最初の数字を数値に自動変換します (数字から始まらなければ 0 として扱う).
(http://www.php.net/manual/ja/language.types.string.php#language.types.string.conversion)

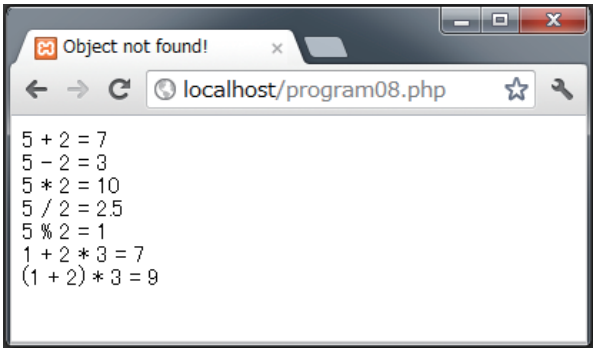
"5 + 2 = " . 5 は "5 + 2 = 5" と言う文字列になり ,
"5 + 2 = 5" + 2 の計算が行なわれるときに ,5 + 2 と変換されて計算されます .
"5 + 2 = " の文字列を "8 + 2 = " などと置き換えて実行してみると計算結果が変化します .
<?php echo "8 + 2 = " . 5 + 2 ?> は 10 と表示されます .

【簡単解決コース】

エディタで新規作成して ,xampp のドキュメントルートに ”program09.php” というファイル名を付けて保存して下さい .(文字コードは UTF-8：BOM なしです)

図 36 を参考にプログラムを入力して上書き保存し , ブラウザで program09.php にアクセスしてみて下さい (図 30).

図 30 program09.php の実行結果



文字列と数式を結合するときは , 問題が起きないように , かならず数式を括弧で括るようにしたほうが安全です .

比較演算子

PHP で使える演算子は数値計算と文字列結合だけではありません . ある条件が正しいか正しくないかを判断する , 比較演算子というものがあります .

比較演算子では数値の大小や 2 つのデータが等しいかどうかを判別し , 論理型 (Boolean) の結果を返します . 論理型とは ,TRUE(真) または FALSE(偽) の二つの値を取ります .

例えば 10 > 5 (">" が比較演算子) は 10 は 5 よりも大きいので結果は TRUE になり ,5 > 10 は 5 は 10 より大きく " ない " ので結果は FALSE になります .

比較演算子には左辺の値が右辺以上 , 以下 , より大きい , より小さい を表わす 4 種類と , 左辺と右辺が等しいか等しくないかの 2 種類があります .

また , 等しいか等しくないかはさらにそれぞれ 2 種類あり ,
データの型が同じで等しい
データの型を問わず (変換すれば) 等しい
データの型が違うか , 値が等しくない
データの型を問わず ,(変換しても) 値が等しくない
の計 4 種類になります .

図 37 program10.php では , 各パターンの例を挙げています . 結果は図 31 のようになります .

図 31 program10.php の実行結果

1 < 2 → bool(true) (左辺が小さいので TRUE)
1 > 2 → bool(false) (左辺が小さいので FALSE)
2 < 1 → bool(false) (左辺が大きいので FALSE)
2 > 1 → bool(true) (左辺が大きいので TRUE)
2 <= 2 → bool(true) (左辺が右辺以下なので TRUE)
2 >= 2 → bool(true) (左辺が右辺以上なので TRUE)
1 == "1" → bool(true) (値が等しいので TRUE)
1 == "2" → bool(false) (値が違うので FALSE)
1 === "1" → bool(false) (型が違うので FALSE)
1 === 1 → bool(true) (型も値も等しいので TRUE)
1 != "1" → bool(false) (値が等しいので FALSE)
1 != "2" → bool(true) (値が違うので TRUE)
1 !== "1" → bool(true) (型が違うので TRUE)
1 !== 2 → bool(true) (値が違うので TRUE)
1 !== "2" → bool(true) (値も型も違うので TRUE)
1 !== 1 → bool(false) (値も型も等しいので FALSE)

論理型は文字列に変換されると TRUE は "1",FALSE は ""(空文字列) になります . そのため echo では ,"TRUE","FALSE" とは表示されず ,"1" または "" と表示されます . 論理型を正確に表示する場合には ,var_dump などを使う必要があります .

PHP では , 論理型だけではなく , 他のデータ型も論理型と見做されることがあります .

FALSE と見做されるもの

- 論理型の FALSE
- 整数型の 0
- 実数の 0.0
- 空文字列 "" または ''
- 文字列型の "0" または '0'
- 要素の数が 0 の (空の) 配列
- NULL
- 値がセットされていない変数

TRUE と見做されるもの

- FALSE と見做されるもの以外の全て

論理演算子

論理演算子とは論理型の否定や , 論理型の合成に使用される演算子です .
日本語でも , 〜ではない (not), 〜かつ〜 (and), 〜または〜 (or) などがありますが ,PHP では主に記号 (!,&&,||) で表します . (表 2)

表 2
否定 (not " 〜ではない ")

値	! 値
TRUE	FALSE
FALSE	TRUE

論理積 (and " 〜かつ〜 ")

左辺	右辺	左辺 && 右辺
TRUE	TRUE	TRUE
TRUE	FALSE	FALSE
FALSE	TRUE	FALSE
FALSE	FALSE	FALSE

論理和 (or " 〜もしくは〜 ")

左辺	右辺	左辺 || 右辺
TRUE	TRUE	TRUE
TRUE	FALSE	TRUE
FALSE	TRUE	TRUE
FALSE	FALSE	FALSE

条件分岐

命令を並べただけではプログラムは毎回同じことしか出来ません .Web アプリケーションでは , 入力されたデータや , 実行された時間などによって動作を切り替えて欲しいものです .

そのために使用するのが条件分岐という機能です . 条件分岐は , 先に説明した比較演算子の結果や , 変数の値などによってプログラムの動作を切り替えることができます .

PHP では、条件分岐には主に if 構文を使用します。if (条件) { 条件が真のとき実行されるプログラム } のように記述します。条件は、前の項で解説した、比較演算子や、論理演算子などを組みあわせて記述します。条件の結果として、FALSE(偽) となる値と、TRUE(真) になる値については特に重要ですので憶えておいて下さい。

if 構文で条件分岐を行なうことができますが、ある条件が真の場合は A というプログラムを、そうで無い場合は B と言うプログラムを実行したい場合にはどうしたらよいでしょうか。

if(条件) { A }
if(! 条件) { B }

というように記述しても良いのですが、if-else 構文を使用するともっと簡潔に記述することができます。

```
if( 条件 ) {
  条件が真の場合実行されるプログラム
} else {
  条件が偽の場合実行されるプログラム
}
```

また、

```
if ( 条件 1) {
  条件 1 が真の場合実行されるプログラム
} else if ( 条件 2) {
  条件 1 が偽で条件 2 が真の場合
  実行されるプログラム
}else {
   条件 1、条件 2 ともに偽の場合実行される
}
```

という書き方もできます。

実用的な例として ,program02.php を元に ,POST されたデータが無い場合には , 入力ページに表示を切り替える (リダイレクト) 機能を作ってみましょう .

課題 4

目的：
program11.php がブラウザで直接ひらかれた場合 , 入力ページ (progrma11.html) に切り替える .

出来る事：
if(条件){ 条件が真のとき実行される }
isset(変数) 関数でデータがあるかチェックできる .(データがあれば真 , なければ偽)
isset($_POST['name 属性 ']) で POST リクエストによって要求されたか判別できる
(GET リクエストの場合は isset($_GET['name 属性 ']))
header() 関数 HTTP ヘッダを出力する .
header('Location: URL');
exit();
と書くと指定した URL に遷移する (切り替わる)

入力の章で作成した program02.html,program02.php をコピーして ,program11.html と program11.php にファイル名を変更します .

図 38 program11.html, 図 39 program11.php を参考に書き換えます . ファイルを保存したら , ブラウザで program11.php にアクセスしてください . 入力画面が表示されて , アドレスバーに program11.html と表示されていれば正常に動作しています .

この課題で重要な点は ,! isset($_POST['data']) で "data と言う名前のデータを受け取って**いない**場合 " になるということと ,header 関数でリダイレクトする場合は <html> タグの前に ,header 関数を使う点です .

header 関数を使用する前に HTML などをが出力されてしまうと , 不具合が出る可能性があります .

リクエストで受け取ったデータを処理する場合は全てのデータに対して isset 関数でチェックするようにして下さい .

GET リクエストならアドレスバーで加工したリクエストを簡単に送信できますし ,POST リクエストでもブラウザ以外のツールで加工したリクエストを送信出来ます . リクエストに含まれるデータが完全である保証はありませんので注意して下さい .

応用課題

目的：
課題 4 と同等の動作を 1 つの PHP ファイルで行なう
出来る事：
作例 11 program11.php
HOST 40 program11.php 場合は結果を表示し ,POST リクエストされたデータが無ければ入力画面を表示します .

11. まとめ

URI(URL)

http://example.com/index.html のような、インターネット上のリソースの場所を示す文字列を URL と呼びます .URL は主にスキーム、ホスト名、パス、クエリ文字列から構成されます .

数値

コンピュータは全ての情報を数値として扱います . コンピュータは基本的に 2 進数を使用して動作します .bit や byte などの単位や、2 進数と 10 進数、16 進数の関係と、2 進数や 16 進数での切りの良い桁で表せる数については憶えておくと役に立つことも多いと思います .

文字

コンピュータは各文字に番号を振って、数値の列として文章などを管理します . 日本語などでは、エンコードによって同じ文字でも違う番号が割り振られることに注意してください .

データ型

PHP で扱うデータには , 型があると言うことを説明しました . 文字列型 , 整数型などさまざまな型がありますが , プログラムを記述する際には扱うデータがどのような型なのか意識するようにしてください . また , 処理によっては型の自動変換が発生することがあります . 文字列と数値の結合や比較を行なう際にはどのような変換が行なわれるのか注意してください .

変数

データに名前を付ける方法として変数という機能を使用します .PHP では変数名として ,$ 記号のあとに _ (アンダースコア) や英数字が使えます . 変数にデータ (値) を代入するには , 変数名 = 値とイコール記号を使用します . 数学と違い両辺が等しいと言う意味ではないことに気を付けてください . 逆に変数の内容を取り出す場合は変数名を記述するとプログラムが実行されるタイミングで値に置き替えられます .

演算子

＋や－などの計算に使用される記号のことを演算子と呼びます . 演算子は加減乗除だけではなく , 比較演算子や論理演算子など様々なものがあります . 演算子には優先順位 (+,- より *,/ が優先) や結合方向 (同じ優先順位の演算子が連続しているときに左から処理するか , 右から処理するか) などが決まっています . 演算の順序を変えるには () を使用します .

条件分岐

ある条件が TRUE(真) であるか FALSE(偽) であるかによってプログラムの処理の流れを変更することを条件分岐と言います . 条件で TRUE になる値と FALSE になる値については複数あり , 気を付けていないと想定外の動作することがあります .

図 32　program06.php

```
<html>
<head>
<meta http-equiv="content-type" content="text/html; charaset=UTF-8">
<title>program06: データ型 var_dump</title>
</head>
<body>
<table>
<tr>
<th> 論理値 </th>
<td> <?php var_dump(true); echo ' / '; var_dump(false); ?> </td>
</tr>
<tr>
<th> 整数 </th>
<td><?php var_dump(123); ?></td>
</tr>
<tr>
<th> 実数 </th>
<td> <?php var_dump(45.6); ?></td>
</tr>
<tr>
<th> 文字列 </th>
<td> <?php var_dump(' 文字列 '); ?></td>
</tr>
<tr>
<th> 配列 </th>
<td> <?php var_dump(array('a',123,true)); ?></td>
</tr>
<tr>
<th> オブジェクト </th>
<td> <?php var_dump(new DateTime()); ?></td>
</tr>
<tr>
<th>NULL</th>
<td> <?php var_dump(null); ?></td>
</tr>
</body>
</html>
```

図 33　program07.php

```
<html>
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
  <title>program07: 変数 </title>
 </head>
 <body>
  <?php
    $message = 'PHP プログラミング ';
    echo $message;
  ?>
 </body>
</html>
```

図 34　program08.php

```
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
  <title>program08: 四則演算 </title>
</head>
<body>

 <?php echo 5 + 2  ?><br />
 <?php echo 5 - 2  ?><br />
 <?php echo 5 * 2  ?><br />
 <?php echo 5 / 2  ?><br />
 <?php echo 5 % 2  ?><br />
 <?php echo 1 + 2 * 3 ?><br />
 <?php echo (1 + 2) * 3 ?><br />

</body>
</html>
```

図 35　program09a.php

```
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
   <title>program09a: 文字列結合演算子（不具合あり）</title>
</head>
<body>

 <?php echo '8 + 2 = ' . 5 + 2  ?><br />
 <?php echo '5 - 2 = ' . 5 - 2  ?><br />
 <?php echo '5 * 2 = ' . 5 * 2  ?><br />
 <?php echo '5 / 2 = ' . 5 / 2  ?><br />
 <?php echo '5 % 2 = ' . 5 % 2  ?><br />
 <?php echo '1 + 2 * 3 = ' . 1 + 2 * 3 ?><br />
 <?php echo '(1 + 2) * 3 = ' . (1 + 2) * 3 ?><br />

</body>
</html>
```

図３６　program09.php

```
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
   <title>program09: 文字列結合演算子 </title>
</head>
<body>

 <?php echo '5 + 2 = ' . (5 + 2)  ?><br />
 <?php echo '5 - 2 = ' . (5 - 2)  ?><br />
 <?php echo '5 * 2 = ' . (5 * 2)  ?><br />
 <?php echo '5 / 2 = ' . (5 / 2)  ?><br />
 <?php echo '5 % 2 = ' . (5 % 2)  ?><br />
 <?php echo '1 + 2 * 3 = ' . (1 + 2 * 3) ?><br />
 <?php echo '(1 + 2) * 3 = ' . (1 + 2) * 3 ?><br />

</body>
</html>
```

図 37　program10.php

```
<html>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
  <title>program10: 比較演算子 </title>
<body>
 <?php echo '1 < 2  → '; var_dump(1 < 2); ?> ( 左辺が小さいので TRUE)<br /><br />
 <?php echo '1 > 2  → '; var_dump(1 > 2); ?> ( 左辺が小さいので FALSE)<br /><br />
 <?php echo '2 < 1  → '; var_dump(2 < 1); ?> ( 左辺が大きいので FALSE)<br /><br />
 <?php echo '2 > 1  → '; var_dump(2 > 1); ?> ( 左辺が大きいので TRUE)<br /><br />
 <?php echo '2 <= 2  → '; var_dump(2 <= 2); ?> ( 左辺が右辺以下なので TRUE)<br /><br />
 <?php echo '2 >= 2  → '; var_dump(2 >= 2); ?> ( 左辺が右辺以上なので TRUE)<br /><br />



 <?php echo '1 == \'1\'  → '; var_dump(1 == '1'); ?> ( 値が等しいので TRUE)<br /><br />
 <?php echo '1 == \'2\'  → '; var_dump(1 == '2'); ?> ( 値が違うので FALSE)<br /><br />
 <?php echo '1 === \'1\' → '; var_dump(1 === '1'); ?> ( 型が違うので FALSE)<br /><br />
 <?php echo '1 === 1     → '; var_dump(1 === 1); ?> ( 型も値も等しいので TRUE)<br /><br />
 <?php echo '1 != \'1\'  → '; var_dump(1 != '1'); ?> ( 値が等しいので FALSE)<br /><br />
 <?php echo '1 != \'2\'  → '; var_dump(1 != '2'); ?> ( 値が違うので TRUE)<br /><br />
 <?php echo '1 !== \'1\' → '; var_dump(1 !== '1'); ?> ( 型が違うので TRUE)<br /><br />
 <?php echo '1 !== 2     → '; var_dump(1 !== 2); ?> ( 値が違うので TRUE)<br /><br />
 <?php echo '1 !== \'2\' → '; var_dump(1 !== '2'); ?> ( 値も型も違うので TRUE)<br /><br />
 <?php echo '1 !== 1     → '; var_dump(1 !== 1); ?> ( 値も型も等しいので FALSE)<br /><br />
</body>
</html>
```

図 38　program11.html

```
<html>
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
  <title>program11: 条件分岐 </title>
 </head>
 <body>
  <form method="POST" action="./program11.php">
   <p> 文章 <input type="text" name="data">
   <input type="submit" value=" 送信 "></p>
  </form>
 </body>
</html>
```

図 39　program11.php

```
<?php
 if( ! isset($_POST['data']) ) {
          header('Location: ./program11.html');
                  exit();
 }
?>
<html>
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
  <title>program11: 条件分岐 </title>
 </head>
 <body>
  <?php echo htmlentities($_POST['data'],ENT_QUOTES,'UTF-8') . ' が入力されました '; ?>
 </body>
</html>
```

図 40　program11ex.php

```
<html>
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
  <title>program11ex: 条件分岐 ( 応用課題 )</title>
 </head>
 <body>
 <?php
  if( isset($_POST['data']) ) {
   echo htmlentities($_POST['data'],ENT_QUOTES,'UTF-8') . ' が入力されました ';
  }else{
 ?>
   <form method="POST" action="./program11ex.php">
    <p> 文章 <input type="text" name="data">
    <input type="submit" value=" 送信 "></p>
   </form>
 <?php } ?>
 </body>
</html>
```