EC Intelligence スクリプト利用ガイド

１．スクリプトについて

EC Intelligence では、メールなどの配信コンテンツおよびレコメンドテンプレート機能でスクリプトが利用できます（接客機能では利用できません）。

EC Intelligence のスクリプトは、Apache Velocity というテンプレートエンジンを利用しています。詳細については、下記サイトなど Apache Velocity の情報を参照ください。

https://www.techscore.com/tech/Java/ApacheJakarta/Velocity/index/

２．変数

商品情報を変数として利用できます。

変数名は、「$item.商品情報連携項目」となります。

$item.id
$item.group_id
$item.title
$item.summary
$item.url
$item.shop_id
$item.content
$item.price
$item.img_url
$item.mb_img_url
$item.string1～$item.string20
$item.number1～$item.number5
$item.disp1～$item.disp20

カート商品やお気に入りなどの商品情報は、$item の代わりに下記の変数名を利用します。

#cart
-> $cart_items
#favorite

-> $favorite_items

#custom

-> $custom_items

#access

-> $access_items

#order

-> $order_items

３．その他の変数

$items.size()・・・レコメンドなどで取得された商品数

$foreach.count・・・foreach ループ中のカウント

４．分岐の記述(#if～#end)

if 文を利用して条件分岐の記述が可能です。

#if(条件)

[出力]

[#elseif(条件 2) [出力 2]]

・・

[#else [出力 3]]

#end

条件部分には、==、=、>、<、<=、>=、!などの論理演算子を利用することができます。

if 文に変数を利用して下記のような記述が出来ます。

#if($items.price() >= 1000)

価格が 1000 以上の場合

#if($items.size() >= 3)

レコメンドなどで取得した商品数（$item.size）が 3 件以上の場合

#if($foreach.count ==1)

#foreach で繰り返し処理をする場合に、1 件目の商品の場合

#if($address1.startsWith("大阪府"))

変数 address1 の文字列が“大阪府”で始まる場合

５．繰り返し処理(#foreach～#end)

#foreach は繰り返し処理を行うための指示子です。 オブジェクトのリストを使ってループさせます。

```
#foreach($item in $items)
  $item.id
#end
```

取得した商品リストを繰り返し処理で、商品 ID を出力します。

#foreach～#end の中で今何番目の繰り返しかは$ foreach.count 変数で取得できます。

例)

```
#foreach($item in $items)
  $foreach.count：　$item.id
#end
```

1 : itemid1

2 : itemid2

3 : itemid3

$foreach.first：ループの最初の要素である場合、true

$foreach.last：ループの最終の要素である場合、true

6．コマンド

#stopSend：送信を停止します。例えば、取得した商品数が 2 件以下の場合は送信しない

例）

```
#if($items.size() <= 2)
    #stopSend
#else
#end
```

#set：値を代入します。

例）

```
#set($cnt1=0)
#foreach($item in $cart_items)
#set($cnt1=$cnt1+1)
#end
```

5．ユーティリティ関数

- util.abbr(文字列,文字数,省略文字)

  ${util.abbr("abcdefghijklmn", 5, "・・・")}

  文字列を途中で省略します。

引数は文字列（変数が利用できます）、表示する文字数、後ろにつける文字列（定義しない場　合は、...）

- $success_count

ファイル機能で利用いただけます。配信に成功したカウントになります。

例えば、ファイル機能で出力するファイルにヘッダ行を付けたい場合、下記のように、成功したレコード数が 0 の時（最初の会員）の時にヘッダ行の文字列を出力します。

```
#if ($success_count == 0)
会員番号,姓,名
#end
$customer_key,$last_name,$first_name
```

- util.number(value)：カンマ区切数値フォーマットで変換します。
${util.number('3000')} -> 3,000　${util.number(${item.price})}

- util.date(value)：日付フォーマットで変換します。

${util.number('20170101')} -> 2017-01-01　　${util.date(${number1})}

- util.url(url, parameter)：url とパラメータを結合します。

${util.url('http://www.scinable.net', 'string1=1&string1=2')} ->

http://www.scinable.net?string1=1&string1=2

- util.isEmpty(value)：value の値がないと true を返します。
#if($util.isEmpty(${item.price}))

- util.isNotEmpty(value)：value の値があると true を返します。
#if($util.isNotEmpty(${item.price}))

- $util.toInt(文字列) : 文字列を int 型の数字に変換します。数字の計算に使用されます。

- $util.toLong(文字列) : 文字列を long 型の数字に変換します。数字の計算に使用されます。

- $util.nowDate() : 現在の日付を日付計算可能な型で返します。

$util.nowDate().plusDays()：現在の日付から plusDate で指定した日数後の日付を返します。

- $util.toDate(yyyyMMdd 形式の文字列) : 引数の日付を日付計算可能な型に変換して返します。

- $util.nowDateTime() : 現在の日時を日時計算可能な型で返します。

- $util.toDateTime(yyyyMMddHHmmss 形式の文字列) : 引数の日時を日時計算可能な型に変換して返します。

- $util.betweenDays(開始日,終了日) : 開始日から終了日まで何日すぎたかを返します。

両引数には$util.nowDate()・$util.toDate()・$util.nowDateTime()・$util.toDateTime()から得た値を指定します。

例)util.nowDate()は 20181011 と仮定。

#set($days = ${util.betweenDays(${util.toDate("20180905")},${util.nowDate()})}) # 20180905 から nowDate(20181011)までは 36 日になる。

#set($diff = 120 - ${days}) # 残りの日数を計算。120 - 36 = 84 日になる。

あと${diff}日

=> あと 84 日

- $util.betweenHours(開始日時,終了日時) : 開始日時から終了日時まで何時間すぎたかを返します。

両引数には$util.nowDateTime()や$util.toDateTime から得た値を指定します。

- $util.toDateTimeCustom(日付を表す文字列,日付のパターン) : 特定パターンの日付文字列を日付計算可能な型に変換して返します。

- $util.formatter(パターン) : 日付計算可能な型を逆に特定パターンの文字列に表現する時、使用される関数です。

6. 日付の計算および指定例

✓ $util.nowCalc(式,出力結果パターン)
現在の日時に対し、式を実行し、その結果を出力パターンで加工して返します。
引数
式
単位 1:増減量 1,...単位 x:増減量 x
単位は日時の要素です。y は年、m は月、d は日です。
増減量には+-する数字を指定します。日(d)に限り、s や e を指定できます。s は最初日(1 日)を、e は最終日を取得することを意味します。

例 1)m:1 -> 翌月の同日(月を+1 するため)
例 2)m:-1 -> 先月の同日(月を-1 するため)
例 3)m:1,d:1 -> 翌月の翌日(月を+1 し、日を+1 するため)
例 4)m:-1,d:e -> 先月の最終日(月を-1 し、最終日の e を指定したため)
出力結果パターン
結果として取得する日時のパターンを指定します。
例 1)yyyyMMdd -> 年月日
例 2)dd -> 日
サンプル
$util.nowCalc("m:1,d:1","dd") => 現在が 2021/11/19 なら、20
$util.nowCalc("m:1,d:s","dd") => 現在が 2021/11/19 なら、01
$util.nowCalc("m:1,d:e","dd") => 現在が 2021/11/19 なら、31

✓ $util.dateCalc(日時文字列,日時文字列のパターン,式,出力結果パターン)
計算を行う対象(日時文字列、日時文字列のパターン)を引数に指定できる以外は$util.nowCalc と同じです。
サンプル
$util.dateCalc('20211119','yyyyMMdd','m:1,d:e','yyyyMMdd') => 20211231
$util.dateCalc('20211119','yyyyMMdd','m:1,d:s','dd') => 01
$util.dateCalc('20211119','yyyyMMdd','m:1,d:e','dd') => 31
$util.dateCalc('20211119','yyyyMMdd','m:1','yyyyMMdd') => 20211219
$util.dateCalc('20211119','yyyyMMdd','m:1','dd') => 19
$util.dateCalc('20211119170125','yyyyMMddHHmmss','m:1','yyyyMMddHHmmss') => 20211219170125
$util.dateCalc('20211119','yyyyMMdd','m:-1,d:e','yyyyMMdd') => 20211031
$util.dateCalc('20211119','yyyyMMdd','m:-1,d:e','dd') => 31
$util.dateCalc('20211119','yyyyMMdd','m:-1','yyyyMMdd') => 20211019
$util.dateCalc('20211119','yyyyMMdd','m:-1','dd') => 19
$util.dateCalc('20211119170125','yyyyMMddHHmmss','m:-1','yyyyMMddHHmmss') => 20211019170125

✓ 今日の日付を商品データ差し込み条件に指定
#cart("0","0","1","","","0","0","","","",3,"0","0","","","",0,"",{"number1":[$util.now("yyyyMMdd")]}) <!--#* yyyyMMdd は日付データ形式に合わせます *#-->

7．URL パラメータの記述

URL にパラメータを付ける際は、下記のように記述お願いします。

<a href="${util.url(${item.url},'utm_source=email・・・')}" target="_blank">

8．文字列の分割

変数の文字列が一定文字数以上の場合に、分割して出力する記述です。

```
#macro(split50 $str)
#set($partA=$str)
#set($partB="")
#if ($str.length() > 50)
#set($partA=$str.substring(0, 50))
#set($partB=$str.substring(50))
#end
#end
```

上記を設定したうえで、#split50($dim5)と呼び出すと dim5 が 50 文字以上の場合に、$partA、$partB と分けて利用することができます。