Справочник по WMLS

Версия 1.0 *29.12.2005*

Содержание

Внимание! Данное руководство рассчитано на пользователя, хотя бы немного знакомого с программированием в среде WML, и содержит только основную информацию о скриптах WMLS. Более полное описание с большим количеством примеров можно найти в интернете на по адресу http://www.devguru.com/Technologies/wmlscript/quickref/wmls_intro.html на английском языке.

1. Начало

WMLS - Wireless Mark-up Language Script - скрипт, расширяющий возможности языка WML. Поскольку все операции, указанные в скрипте, выполняются в телефоне пользователя, серверной поддержки на программном уровне он не требует.

WMLS используется для различных операциий с текстовыми и цифровыми переменными, а значит все операции с текстом, а также математические вычисления как раз и выполняются посредством скрипта. Кроме того, в WMLS существуют методы доступа в интернет (только чтение информации) и генерация случайных чисел.

Иногда говорят, что WMLS для WML то же, что и JavaScript для HTML (структура языков действительно сходна). Это не совсем так. Дело в том, что если во втором случае код скрипта часто вставляется в саму страницу HTML, то WMLS - всегда отдельный от WML файл, который на всех телефонах (кроме Nokia, насколько до сих пор известно) вызывается ТОЛЬКО с соответсвующей WML-страницы. Кроме того, в WMLS нет операций с графикой, чем JavaScript изобилует.

Тем, кто ранее имел дело с программированием на C++ и, конечно, JavaScript, будет проще разобраться с WMLS. Структуры языков имеют общие основы.

2. Структура скрипта WMLS

2.1. Использование

Ссылка на WMLS-скрипт выглядит так:

your_script.wmls#your_function(variables)

где your_script - имя скрипта, function - используемая функция, variables - переменные WML, которые необходимо перенести в WMLS; если их несколько, то они разделяются запятыми. Если переносить переменные не требуется, скобки после названия функции следует оставить пустыми.

2.2. Структура скрипта

Любой WMLS скрипт всегда имеет одну или несколько макро-функций. Поэтому скрипт всегда начинается так:

extern function your_function(variables)

где your_function - название выполняемой функции (указываемой в ссылке на скрипт), variables - переменные, переносимые в WMLS из WML (они также указываются в ссылке, а если не указываются, то скобки нужно оставить пустыми), знак $ перед именем переменной уже не требуется. Есть и иной способ, на мой взгляд, более предпочтительный, для переноса переменных из WML в WMLS. О нем будет рассказано позже.

Далее следует код макро-функции, который всегда заключается в фигурные скобки { код }. Внутри кода также могут встречаться участки, заключенные в фигурные скобки. Это операции, выполняемые по какому-либо определенному условию, либо циклы.

Возможен доступ к другим функиям из данной внутри одного WMLS скрипта. Для этого необходимо просто назвать функцию. Например, так:

```
extern function main()
{ ...
var result = calculate();
... }
extern function calculate()
{ ... }
```

Каждая выполняемая операция всегда заканчивается точкой с запятой ; кроме случаев с использованием условий, но об этом позже.

Переменные, используемые в WMLS, пишутся без знака $. Очень важно запомнить, что для работы с переменной ее ВСЕГДА необходимо предварительно объявить, иначе для скрипта она просто не будет существовать, и это, конечно, вызовет ошибку. Переменная объявляется только один раз, потом var перед ее именем писать не нужно. Задавать/объявлять переменную нужно так:

```
var your_variable = data;
```

или так:

```
var your_variable;
```

где your_variable - имя переменной, которое не может совпадать с именами функций, указаний или библиотек; data - значение переменной. Если значение не числовое, а является строкой текста, то его необходимо заключить в кавычки. Вообще строки текста в WMLS всегда заключаются в кавычки.

Обратите внимание, что если в первом случае переменной присваивалось некоторое значение, то во втором она просто объявлялась.

Переменные WMLS используются только самим скриптом, то есть они никак не связаны с WML-переменными, поэтому результаты операций с ними потом необходимо перенести в WML.

Операции над объектами, выполняемые в WMLS, задаются так:

```
Library.function(data);
```

где Library - библиотека (основной оператор), function - функция, data - данные, над которыми производится некоторая операция. Если необходимо совместить несколько операций, то точку с запятой нужно ставить только после основной, например, так:

```
...
var partoftext = String.subString(WMLBrowser.getVar("text"), 0, String.length(len));
...
```

Об указаниях (условиях) и операторах в WMLS читайте в соответсвующих разделах.

Очень важно обратить внимание на то, что WMLS различает индексы букв (малые и большие буквы). Например, если вы напишете WMLBROWSER.Prev(); вместо WMLBrowser.prev() то это, несомненно, будет считаться ошибкой.

Важно также помнить, что в WMLS нет поддержки кириллицы и специальных символов, поэтому русский текст, если он нужен, необходимо переносить из WML.

3. Функции

Функции библиотек WMLS - это уточняющие указания для данной выполняемой операции. Используются только совместно с библиотеками и никогда - отдельно. Об использовании каждой из функций читайте в разделе "библиотеки".

4. Библиотеки

Библиотеки - это ключевые операторы языка WMLS, позволяющие выполнять разнообразные операции. Функции же уточняют, какие именно.

Существует шесть стандартных библиотек.

4.1. Библиотека Dialogs

Dialogs.alert(message);

Выводит на экран телефона сообщение message.

Dialogs.confirm(message, okmessage, cancelmessage);

Просит пользователя подтвердить действие. Возвращает true если пользователь подтвердил действие, иначе false. message - сообщение о выборе, okmessage и cancelmessage - сообщения, появляющиеся на софт-клавишах телефона, означающие соответсвенно "подтвердить" и "отменить".

Dialogs.prompt(message,defaultinput);

Просит пользователя ввести некоторые данные. message - сообщение, defaultinput - значение, которое изначально появляется в форме (т.е. начальное значение).

4.2. Библиотека Float

Float.ceil(value);

Округляет число с плавающей точкой до ближайшего большего целого числа. value (здесь и далее) - некоторое значение.

Float.floor(value);

Округляет число с плавающей точкой до ближайшего меньшего целого числа.

Float.int(value);

Возвращает целую часть числа с плавающей точкой.

Float.maxFloat();

Возвращает наибольшее число с плавающей точкой, поддерживаемое браузером телефона.

Float.minFloat();

Возвращает наименьшее число с плавающей точкой, поддерживаемое браузером телефона.

Float.pow(value, power);

Возводит число с плавающей точкой или целое число в степень power.

Float.round(value);

Округляет число с плавающей точкой.

Float.sqrt(value);

Возвращает арифметический квадратный корень числа.

4.3. Библиотека Lang

Lang.abort(errorDescription);

Прерывает выполнение кода и передает управление вызвавшему. errorDescription - описание ошибки (сообщение).

Lang.abs(value);

Возвращает абсолютное значение (выражение по модулю).

Lang.characterSet();

Возвращает поддерживаемый набор символов.

Lang.exit();

Прекращает выполнение кода и передает управление вызвавшему.

Lang.float();

Проверяет, есть ли поддержка исчисления с плавающей точкой.

Lang.isFloat(value);

Проверяет, возможен ли перевод данного выражения в число с плавающей точкой.

Lang.isInt(value);

Проверяет, возможен ли перевод данного выражения в целое число.

Lang.max(value1, value2);

Возвращает большее из двух данных чисел value1 и value2.

Lang.maxInt();

Возвращает наибольшее распознаваемое целое число.

Lang.min(value1, value2);

Возврщает меньшее из двух данных чисел value1 и value2.

Lang.minInt();

Возвращает наименьшее распознаваемое целое число.

Lang.parseFloat(value);

Конвертирует строку текста в число с плавающей точкой.

Lang parseInt(value);

Конвертирует строку текста в целое число.

Lang.random(value);

Возвращает случайное положительное целое число из множества [0; value].

Lang.seed(value);

Инициирует псевдо-случайную последовательность чисел. value - номер, задающий эту последовательность.

4.4. Библиотека String

String.charAt(string, index);

Возвращает символ на позиции index в строке текста string.

String.compare(string1, string2);

Сравнивает две строки текста и возвращает 1, 0 и -1 соответсвенно если в лексикографическом отношении двух строк первая выше оценена, строки эквивалентны или первая строка оценена ниже второй.

String.elementAt(string, index, separator);

Возвращает элемент строки текста, начиная с элемента index до следующего сепаратора (разделителя). Например, *String.elementAt("This is a test",3," ");* вернет "test". В этом примере в качестве разделителя использован пробел " ". Индекс нумеруется с нуля.

String.elements(string, separator);

Возвращает число элементов в строке текста.

Sting.find(string, subString);

Возвращает позицию указанной подстроки subString в строке текста string.

String.format(format, string);

Форматирует данную строку согласно указанному формату.

Виды фомата String.format(string, format)

Указатель формата - это текстовая строка (поэтому ее нужно заключать в кавычки!) , содержащая данные об обработке текста/числа.

Указатель имеет вид:

%width.precision type text

Описание параметров:

type (необходим для указания):

d - задает положительное или отрицательное целое число (включая ноль).
f - задает положительное или отрицательное число с плавающей точкой (включая ноль).
s - задает строку текста.

width (может быть не указан; если указан, должен быть целым положительным числом, не равным нулю):

Ширина (минимальный размер) числа/строки текста. Если указывается величина, большая фактической длины текста, то слева добавляется соответсвующее число пробелов.

precision (может быть не указан; если указан, должен быть целым положительным числом):

для d - указывает минимальное число цифр в возвращаемом целом числе; если длина строки, содержащей число, меньше указанного значения параметра, то слева от числа добавляется соответствующее число нулей (либо пробелов, зависит от браузера).

для f - указывает точность десятичной дроби, то есть то количество цифр, которое указывается слева от запятой; если необходимо, дробь
округляется либо справа добавляется соответсвующее число нулей.

для s - указывает количество возвращаемых символов строки текста.

text (может быть не указан):

дополнительный текст.

Примеры указателя формата

Указатель формата	Форматируемое значение	Возвращаемый результат
"%4d"	41	41
"%4d"	-41	-41
"%7.5.d"	41	00041
"%6.4f"	12.4012352	12.4012
"Number is = %6.3f"	401.5099	Number is 401.510
"%4.0f kilometers"	987.654	988 kilometers
"%7s"	manfred	manfred
"%7s Mann is a musician"	Manfred	Manfred Mann is a musician
"%6.1f"	John	invalid

Для того, чтобы в указателе формата использовать символ %, пишите %%.

String.insertAt(string, element, index, separator);

Вставляет в строку текста string элемент и разделитель.

String.isEmpty(string);

Проверяет, пуста ли строка string.

String.length(string);

Возвращает длину строки текста.

String.removeAt(string, index, separator);

Удаляет указанную подстроку из строки текста.

String.replace(string, oldSubString, newSubString);

Заменяет в строке текста string подстроки oldSubString на newSubString.

String.replaceAt(string, element, index, separator);

Заменяет в строке текста указанную подстроку на новую.

String.squeeze(string);

Удаляет из строки лишние пробелы.

String.subString(string, startIndex, length);

Создает новую строку текста из существующей по выделенному фрагменту с началом startIndex и длиной length.

String.toString(value);

Конвертирует данное значение value в строку текста.

String.trim(string);

Удаляет пробелы (если они есть) до и после фактической строки текста.

4.5. Библиотека URL

Обратите вниманме на синтаксис адреса URL:

scheme://host:port/path;parameters$query#fragment

URL.escapeString(string);

Заменяет специальные символы в строке текста string на их шестнадцатиричные эквиваленты вида %hh, где hh варьируется от 00 до FF.

URL.getBase();

Возвращает абсолютный текущий URL-адрес WMLS-скрипта.

URL.getFragment(url);

Возвращает фрагмент адреса URL.

URL.getHost(url);

Возвращает хост адреса URL.

URL.getParameters(url);

Возвращает параметры адреса URL.

URL.getPath(url);

Возвращает путь адреса URL.

URL.getQuery(url);

Возвращает запрос URL-адреса.

URL.getReferer(url);

Возвращает наименьший относительный адрес URL.

URL.getScheme(url);

Возвращает вид протокола передачи данных адреса URL.

URL.isValid(url);

Проверяет правильность ввода URL-адреса.

URL.loadString(url, contentType);

Загружает данные из текстового файла по адресу URL на основе вида данных contentType. О видах данных MIME-type читайте здесь.

Виды данных MIME-type (для текстовых файлов)

Вид текстового файла	MIME-Type
Обычный текстовый файл	text/plain
XML-файл	text/xml
HTML-файл	text/html
WML-файл	text/vnd.wap.wml
WMLS-файл	text/vnd.wap.wmlscript

URL.resolve(baseUrl, embeddedUrl);

Возвращает абсолютный адрес URL по базовому и относительному адресам.

URL.unescapeString(string);

Заменяет шестнадцатиричные данные вида %hh на специальные символы, соответсвующие им.

4.6 Библиотека WMLBrowser

WMLBrowser.getCurrentCard();

Возвращает абсолютный текущий URL-адрес WMLS-скрипта.

WMLBrowser.getVar(name);

Возвращает значение WML-переменной с именем name.

WMLBrowser.go(url);

Приказывет браузеру телефона перейти по адресу url.

WMLBrowser.newContext();

Очищает контекст (кэш) браузера телефона.

WMLBrowser.prev();

Приказыает браузеру вернуться на предыдущую карту.

WMLBrowser.refresh();

Заставляет браузер обновить текущую карту.

WMLBrowser.setVar(name, value);

Устанавливает новую (или обновляет старую) переменную WML с именем name, присваивая ей значение value.

5. Указания

Указания - это операции, выполняемые в WMLS-скрипте. Указания не производят операций над отдельными объектами, а отвечают за работу скрипта в целом.

break

Прерывает код в цикле, образованном указаниями for или while, и передает управление коду, находящемуся непосредственно после цикла.

continue

Преостанавливает код в цикле, образованном указаниями for или while, и передает управление коду, проверяющему условие протекание цикла. Непоредственного выхода из цикла continue не вызывает.

for (counting_variable; condition; inc_or_dec) { code }

Задает цикл так, что код code повторяется до тех пор, пока выполняется условие condition. counting_variable - переменная, использующаяся для счета, а inc_or_dec - операция, выполняемая над переменной счета. Например, такой цикл:

for (var count=0; count < 99; count++;) { ... }

if (condition) { code }

Указание условия. Код code выполняется только в том случае, если условие condition верно. После { code } возможно указание else { code1 }, которое означает, что если условие не было верным и первый код не выполнялся, то следует выполнить код { code1 }. Например, так:

```
...
if (numb != -15) { var page = "1"; } else { var page = "2"; }
WMLBrowser.go("page"+page+".wml');
...
```

return

Указание используется для немедленного возврата из функции, причем функции присваивается ее возвращаемое значение. Пример:

```
...
var result = findrandom();
...
extern function findrandom()
{
return Lang.random(500);
}
```

var

Используется для объявления и присвоения значения переменной. Переменная объявляется только один раз, после этого ее нужно использовать без указания var. Например:

```
...
var myname = "Alex";
var myage;
myage = Lang.random(100);
var sentence = "My name is "+myname+" and today I'm "+myage+" years old! =)";
Dialogs.alert(sentence);
...
```

while (condition) { code }

Создает цикл так, что код code выполняется до тех пор, пока справедливо условие condition. Например:

```
...
while (time < 100)
{ time = time + Lang.random(10); }
...
```

6. Операторы

6.1. Арифметические

Арифметические операторы используются для простых и комплексных математических операций.

+

Используется для сложения целых и чисел с плавающей точкой, а так же для соединения двух или более строк текста.

-

Используется для вычитания целых и чисел с плавающей точкой.

*

Используется для умножения целых и чисел с плавающей точкой.

/

Используется для деления чисел с плавающей точкой.

div

Используется для деления целых чисел. Результат всегда округляется (до нуля).

%

Используется для вычисления модуля (абсолютного значения) выражения.

++

Используется для приращения целочисленного значения на единицу.

--

Используется для вычитания целочисленного значения на единицу.

Следующие арифметические операторы используются для операций с битами. Они могут понадобится только программистам с соответсвующим опытом их использования.

Битовые арифметические операторы

Оператор	Битовая операция
<<	Left shift
>>	Right shift
>>>	Zero fill right shift
&	AND
|	OR
^	XOR
~	NOT

6.2. Присвоения

Эти операторы используются для присвоения переменной определенного значения.

=

Простейший оператор присвоения. Пример использования:

var my_variable = a;

Остальные операторы присвоения – это комбинации знака равенства со стандартными операторами (используются для сокращений).

Операторы присвоения, применяемые для сокращенного написания операций

Сокращенное написание	Выполняемая операция
a **+=** b	a = a + b
a **-=** b	a = a – b
a ***=** b	a = a * b
a **/=** b	a = a / b
a **div=** b	a = a div b
a **%=** b	a = a % b
a **<<=** b	a = a << b
a **>>=** b	a = a >> b
a **>>>=** b	a = a >>> b
a **&=** b	a = a & b
a **|=** b	a = a | b
a **^=** b	a = a ^ b

6.3. Сравнения

Операторы позволяют сравнивать различные значения между собой. Исходя из результата, можно использовать эти операторы для подтверждения выполнения каких-либо операций.

Операторы сравнения

Оператор	Значение
==	Равно
!=	Не равно
>	Больше
<	Меньше
>=	Больше или равно
<=	Меньше или равно

Примеры использования:

...
var res = a == b;
...

if (a != b) { ... }
...
while (a < b) { ... }
...

6.4. Условия

Оператор условия используется для присвоения выражению одного из двух возможных значений, выбор основан на булическом тесте условного значения.

expression = condition ? value1 : value2;

expression – это выражение, которому присваивается значение:

value1, если условие condition является true;

value2, если условие condition является false или invalid.

Например:

...
var cond = a == b;
var expr = cond ? "A equals B" : "A doesn't equal B";
Dialogs.alert(expr);
...

6.5. Логические

Операторы используются для выполнения булических тестов над выражениями.

&&

Логический оператор AND (И). Если оба выражения истинны, возвращает true. Если одно из двух выражений ложно, возвращает false. Если любое выражение из двух является invalid, возвращает invalid.

||

Логический оператор OR (ИЛИ). Если одно из двух выражений истинно, возвращает true. Если оба выражения ложны, возвращает false. Если любое выражение из двух является invalid, возвращает invalid.

!

Логический оператор NOT (ОТРИЦАНИЕ). Применяется для произведения отрицания булического выражения true или false. Таким образом, true становится false, а false становится true.

Пример использования логических операторов:

...
var result = (a && b) || (a && !b);
...

6.6. isvalid

Оператор проверяет правильность выражения. Возвращает true, если выражение допустимо, и false, если выражение расценено, как invalid.

Пример:

```
...
var a = 1/0;
var validity = isvalid(a);
Dialogs.alert("The validity of the value appears to be "+validity);
...
```

6.7. string

Операторы типа string применяются при работе со строками текста.

+

Оператор используется для слияния двух строк текста. Вторая строка присоединяется к концу первой.

+=

Используется для не посредствеeного присоединения второй строки текста к первой.
Пример:

```
...
var string1 = "I ";
var string2 = "Am ";
var string3 = "The Man";
var newstring = string1 + string2 + string3;
...
```

Сравните:

```
...
var addstring = "You are going to";
var newsting = "make it!";
newstring += addstring;
...
```

6.8. typeof

Оператор служит для определения типа выражения. WMLS распознает пять типов выражений, поэтому оператор typeof, в зависимости от выражения возвращает целое число от 0 до 4.

Вид	Возвращаемый результат
Целое число	0
Число с плавающей точкой	1
Строка текста	2
Булическое значение	3
Неверно (invalid)	4

Пример:

```
...
var val_type = typeof(987.654);
...
```

7. Примеры скриптов

В данном разделе я приведу примеры достаточно простых скриптов вместе с соответсвующим WML-кодом.

Кроме того, я объясню только принцип работы скрипта, но не детальное его описание. Постарайтесь разобраться сами, в этом большой смысл. Только нучившись разбираться в коде произвольно взятого скрипта человек овладевает необходимыми навыками для дальнейшего успешного программирования.

7.1. Калькулятор

Простенький скрипт, позволяющий выполнять элементарные математические операции.

Код WML (*calc.wml*):

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml12.dtd">
<wml>
<card id = "index" title = "Калькулятор">
<p align="left">
Число A:<br/>
<input name = "vala" format = "*n" emptyok = "false"/><br/>
Число B:<br/>
<input name = "valb" format = "*n" emptyok = "false"/><br/>
Операция:<br/>
<select name = "oper" title = "Опреация">
<option value = "0">+</option>
<option value = "1">-</option>
<option value = "2">*</option>
<option value = "3">/</option>
</select>
<br/>
<a href="calc.wmls#calc()">Считать</a>
</p>
</card>
<card id = "result" title = "Результат">
<p align = "center">
Ответ:<br/>
$(result)<br/><br/>
<a href = "#index">Назад</a>
</p>
</card>
</wml>
```

Код WMLS (*calc.wmls*):

```
extern function calc()
{
var val1 = Lang.parseFloat(WMLBrowser.getVar("vala"));
var val2 = Lang.parseFloat(WMLBrowser.getVar("valb"));
var oper = WMLBrowser.getVar("oper");
var result;
if (oper == "0") result = val1 + val2;
if (oper == "1") result = val1 - val2;
if (oper == "2") result = val1 * val2;
if (oper == "3") result = val1 / val2;
WMLBrowser.setVar("result", result);
WMLBrowser.go("#result");
}
```

7.2. Переход на произвольную страницу

Скрипт генерирует элемент случайной ссылки, на которую затем переходит пользователь. В рассмотренном случае переход осуществляется на произвольную карту одной и той же WML-страницы.

Код WML (*goto.wml*):

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.2//EN"
"http://www.wapforum.org/DTD/wml_12.xml">
<wml>
<card id="index" title="Случайный переход">
<p align="center">
Для перехода на произвольную подстраницу нажмите "Вперед!"<br/><br/>
<a href="goto.wmls#go()">Вперед!</a>
</p>
</card>
<card id="page0" title="Страница 0">
<do type="options" name="main" label="Главная"><go href="#index"/></do>
<p align="center">
Вы попали на страницу номер ноль.
</p>
</card>
<card id="page1" title="Страница 1">
<do type="options" name="main" label="Главная"><go href="#index"/></do>
<p align="center">
Вы попали на страницу номер один.
</p>
</card>
<card id="page2" title="Страница 2">
<do type="options" name="main" label="Главная"><go href="#index"/></do>
<p align="center">
Вы попали на страницу номер два.
</p>
</card>
<card id="page3" title="Страница 3">
<do type="options" name="main" label="Главная"><go href="#index"/></do>
```

```
<p align="center">
Вы попали на страницу номер три.
</p>
</card>
</wml>
```

Код WMLS (*goto.wmls*):

```
extern function go()
{
var rand=Lang.random(3);
Dialogs.alert("Page is "+"page"+rand+".wml");
WMLBrowser.go("goto.wml#page"+rand);
}
```

8. От автора

Я познакомился со скриптовым языком WMLS примерно год назад (середина 2004 года), когда изучал WML. Хотелось бы выразить благодарность в связи с этим ***Алексею Сычеву***, чья поддержка в то время оказала мне очень существенную помощь.

Хотелось бы также вспомнить всех тех, с кем мы прокладывали мосты **RuWAP'a** через сомнительный сервис ***WAP41.com***…

В заключение мне осталось только от всего сердца пожелать вам удачи в изучении WMLS. Помните, этот скриптовый язык не требует серверной поддержки, но позволяет сделать достаточно много интересных вещей, выйти за рамки стандартных возможностей WML.

29.12.2005

Алексей Тепляков