J I H O Č E S K Á U N I V E R Z I T A

PEDAGOGICKÁ FAKULTA

KATEDRA INFORMATIKY

WAP A JAZYK WMLSCRIPT

BAKALÁŘSKÁ PRÁCE

PETR MAZANEC

vedoucí diplomové práce

PaedDr. Petr Pexa

ČESKÉ BUDĚJOVICE 2002

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně, pouze s použitím literatury a zdrojů uvedených v části Použité zdroje.

Petr Mazanec

OBSAH

1. ÚVOD

O technologii mobilního přístupu k celosvětové síti Internet se mluví téměř neustále. Tvůrci WAP rozšířili využití mobilních zařízení o další služby, a tím posunuli hranice bezdrátové komunikace na vyšší pomyslný stupínek technického vývoje. Každý uživatel využívající tohoto přístupu na internet si je vědom jeho současných omezení, jako je například kapacita paměti telefonů, malý displej a rozsah vstupu.

Základním kamenem WAPu se stal bezpochyby jazyk WML ušitý mobilním telefonům přímo na míru, který plně dostačuje pro tvorbu prezentací na mobilních telefonech. Je ovšem statický a v určitých situacích nenabízí uživateli dostatečnou pružnost aplikací. Proto jej tvůrci WAPu doplnili o skriptovací jazyk WMLScript. Svojí syntaxí je dosti podobný Javascriptu. Lze jej využít především pro kontrolu uživatelského vstupu, komunikaci se samotným mobilním zařízením a pro přenesení části logiky na klienta, aby se omezila nutnost poměrně pomalé komunikace se serverem.

Mnozí tvůrci internetových stránek jsou dnes nuceni se naučit základní práci s WML. V současné době můžeme najít výborné učebnice tohoto jazyka (jmenuji jeden příklad za všechny http://www.kosek.cz), ale problém skriptování je zde stále dosti opomíjen.

Cílem této práce je podrobně seznámit uživatele s jazykem WMLScript a dalšími způsoby „skriptování“ na WAPu. Také zde chci nabídnout řešení různých problémů při tvorbě dynamických WAPových stránek. Nebudu se již zabývat samotným jazykem WML, neboť tuto problematiku již řešil Ing. Marek Medve ve své Diplomové prácí “WAP a jazyk WML“ z roku 2001. Tuto práci zároveň doporučuji prostudovat jako materiál, který Vás uvede do problému WAPu.

2. WMLScript

2.1 Úvodní slovo k WMLScripu

Jazyk WML, který zahrnuje formátování textu, vkládání obrázků a odkazů, je statický. Není určen k tvorbě dynamických prvků jako jsou například jednoduché výpočty a ověřování správnosti dat. Pro překonání těchto omezení a pro programování funkcí využitelných na wapových zařízeních byl vytvořen skriptovací jazyk WMLScript. Jazyk jenž je založen na ECMAScriptu a svoji konstrukcí se dosti podobá JavaScriptu.

WMLScript je procedurální jazyk podporující své standardní knihovny. Pro zrychlení komunikace s klientem je překládán do bytecodu. Skriptování zvyšuje přehlednost a dobře doplňuje jazyk WML podobně jako tomu je v případě Javascriptu a HTML.

2.2 Základ WMLScriptu

WMLScript je velice jednoduchý a snadno naučitelný skriptovací jazyk. Podobnost s JavaScriptem dává výhodu programátorům webových aplikací, ale ani pro úplného začátečníka by neměl být problém ovládat základní syntaxe.

Všechny skripty se musí zapisovat odděleně od WML kódu do samostatných souborů, které mají obvykle příponu **.wmls**. Soubor obsahuje definice několika funkcí. Některé z těchto funkcí jsou definovány jako externí, což umožňuje odkazovat se na funkce ležící na jiné URL adrese.

2.3 Lexikální stavba

Tato část popisuje souhrn základních pravidel, která specifikují , jak psát programy v jazyce WMLScript.

Komentáře

Jako každý programovací jazyk má i WMLScript možnost komentářů. Které jsou zcela shodné s komentáři Javy či JavaScriptu .

WMLScript má možnost si vybrat ze dvou komentářů:

- Jednořádkový komentář – začíná // a vše, co je za nimi až do konce řádky je komentář:

  var pristup; // proměnná přístup mapuje počet vstupu na stránku

- Víceřádkový komentář – začíná znaky /* a až do výskytu znaků */ je vše komentář

  ```
  /* tato proměnná mapuje počet
     vstupu na danou stránku   */
   var pristup;
  ```

Identifikátory

Identifikátory se používají k pojmenování a popisu prvků WMLScriptu jako je proměnná a funkce. Identifikátor může začít písmenem nebo podtržítkem a nesmí obsahovat speciální znaky, klíčová slova, nebo rezervovaná slova užívaná v jazyce. Při psaní identifikátorů se rozlišují malá a velká písmena.

Tabulka č. 1 - Přehled rezervovaných slov

Přehled rezervovaných slov			
Klíčová slova			
access	equiv	meta	else
agent	extern	name	invalid
break	for	return	path
continue	function	typeof	domain
div	header	use	if
div=	http	user	var
Klíčová slova neužívaná WMLScriptem			
delete	null	new	lib
in	this	with	void
Nevyužitá rezervovaná slova			
case	fefault	finally	sturct
catch	do	import	super
class	enum	private	seitch
const	export	public	throw
debugger	extends	sizeof	try

Deklarace proměnných

Wmlscript je slabě typový jazyk. Obsahuje pouze typ var, který se deklaruje jednoduše pomocí var a názvu proměnné. Var vnitřně podporuje boolean, integer, floating-point, string a speciální typ invalid, kterého je užíváno v případě neplatného datového typu (potřebujeme-li jej oddělit z ostatních vnitřních datových typů). Jazyk nedává možnost nikterak tyto typy označit a automaticky konvertuje na daný typ, který potřebujeme. Přiřazení hodnoty do proměnné se děje pomocí operátoru "=". Do proměnné můžeme přiřadit číslo nebo textový řetězec.

Hodnoty textových řetězců je potřeba uzavřít do uvozovek nebo apostrofů. Proměnná musí být deklarovaná před místem použití, inicializace je nepovinná. Neinicializované proměnné automaticky obsahují prázdný řetězec(" ").

Příklad:

```
var kolobezka = true;   // vnitřně Boolean
var cislo = 12;         // vnitřně Integer
var desetine_cislo  = 37.7;  // vnitřně Float
var cislo_jinak = "XII";     // vnitřně String
var pristup = invalid;       // vnitřně Invalid
```

Platnost proměnné začíná místem deklarace uvnitř funkce a končí spolu s danou funkcí. Ve WMLScriptu nelze definovat globální proměnné.

Typy proměnných

V kapitole o deklaraci proměnných jsem již zmínil, že WMLScript má specifikovaný pouze typ var a vnitřně užívá různé datové typy. Některé z datových typů mají své příslušné knihovny funkcí, jenž velmi usnadňují práci s daty. Tyto funkce jsou volány pomocí jména knihovny, tečky a názvu dané funkce.

V následujících bodech jsem podrobně popsal jednotlivé datové typy:

- ***Logický typ a jeho konstanty***
  - pro logický typ se používá typ boolean o dané velikosti jednoho bitu. Typ boolean může nabývat pouze dvou hodnot, které jsou reprezentovány logickými konstantami true (má hodnotu 1) a false (má hodnotu 0).

- ***Celočíselný typ integer***
  - velikost integeru je 32 bitů. Minimum a maximum můžeme zjistit pomocí knihovní funkce “Lang.minInt()“ a “Lang.maxInt()“. Celočíselné konstanty mohou být zapsány ve třech číselných soustavách:

- Desítkové(dekadické) – posloupnost číslic 0 až 9, z nichž první nesmí být 0.
- Osmičkové(oktalové) – číslice 0 (nula) následovaná posloupností osmičkových číslic 0 až 7.
- Šestnáctkové(hexadecimální) – číslice 0 (nula) následovaná znakem x (nebo X) a posloupností šestnáctkových číslic 0 až 9, a až f nebo A až F.

Příklad:

```
var x = 86;
var y = 0126; // v osmičkové soustavě
var z = 0x56; // v šestnáctkové soustavě
```

- ***Reálný datový typ (floating-point)***
  - velikost je stejná jako u integeru 32 bitu. Minimum a maximum můžeme opět zjistit pomocí zápisů funkcí Float.minFloat() a Float.maxFloat(). Reálné konstanty se tvoří podle běžných zvyklostí. Mohou začínat a končit desetinou tečkou, obsahovat znaménka, exponenty atd.

Příklad:

```
var x = 99.7;
var y = .7   // stejně jako 0.7
var z = +10.; // stejně jako 10.00
```

- ***Řetězce (String)***
  - řetězce ze zapisují do uvozovek jednoduchých (‘ ‘), nebo dvojitých (“ “). Řetězce mohou obsahovat číslice, písmena a speciální znaky. WMLScript podporuje speciální escape

sekvence jenž se vyskytují uvnitř řetězců (viz následující tabulka č.2).

Tabulka č. 2 – Escape sekvence

Escape sekvence			
sekvence	unicode	význam	symbol
\'	\u0027	apostrof (single quote)	'
\"	\u0022	uvozovky (quote)	"
\\	\u005C	zpětné lomítko (backslash)	\
\/	\u002F	lomítko (slash)	/
\b	\u0008	Posun doleva (backspace)	
\f	\u000C	Form feed	
\a	\u000A	nová řádka (newline, linefeed - LF)	
\r	\u000D	návrat na začátek řádky (carriage return - CR)	
\t	\u0009	tabulátor (tab - HT)	

WMLScript umožňuje automatické zřetězování dlouhých literálů oddělených mezerami, tabelátory, nebo novými řádkami, před kterými ale musí být znak “+“, nebo “+=“.

Délka řetězce se zjistí zápisem funkce String.length().

Příklady na řetězce:

```
var pozdrav = „Ahoj“;
var delka = String.length(pozdrav); // vrátí  hodnotu 4
var pozdrav +=  „ Pepo“ + „ a“ + „ Aleno“;
//proměnná obsahuje řetězec „Ahoj Pepo a Aleno“
```

- ***Typ Invalid***
  - WMLScript podporuje typ invalid, který vyjadřuje neplatnou hodnotu. Hodnotu dostaneme například při dělení nulou.

Ke kontrole typu daného výrazu využíváme operátor „**isvalid**“. Jestliže typ výrazu je nesprávný vrací booleanovskou hodnotu false, v opačném případě vrací boolenovskou hodnotu true.

Příklad:

```
var bluma = 10;
var OK = isvalid(bluma); // vrátí true
var KO = isvalid(bluma/0); // vrátí false
```

Operátor typeof

Operátor typeof vrací hodnotu popisující datový typ daného výrazu (neprovádí se žádná konverze). Typ integer vrátí 0, floating-point 1, string 2 , boolean 3 a invalid 4.

Příklad:

```
Var bluma = 10;
Var hodnota = typeof bluma; // vrátí 0 (integer)
Var pokus = 10/0;
Var hodnota2 = typeof(pokus); // vrátí 4 (invalid)
```

Automatické datová konverze

V některých případech funkce WMLScriptu vyžadují určitý druh vstupních dat. WMLScript podporuje automatické datové konverze, aby vyhověl požadavkům operátorů. Proměnné nemají specifikovaný typ, nicméně vnitřně užívají datové typy popsané v předchozí kapitole.

Každý operátor podporuje určitou množinu předefinovaných typů. Jestliže operand neobsahuje správný typ, provede se automatická konverze.

Legální datové konverze jsou vyjádřeny v následující tabulce:

Tabulka č.3 – Automatická konverze datových typů

Automatická konverze datových typů				
konverze na typ	*boolean*	*integer*	*floating-point*	*string*
boolean true	-	1	1.0	"true"
boolean false	-	0	0.0	"false"
integer 0	false	-	0.0	řetězec "0"
integer různý od nuly	true	-	číslo + desetiny	daný řetězec čísel
floating-poit 0.0	false	nelze	-	řetězec "0.0"
floating-poit různý od nuly	true	nelze	-	daný řetězec čísel
string prázdný ("")	false	nelze	nelze	-
string neprázdný	true	danou hodno-tu ve tvaru integer	danou hodnotu ve tvaru float. Point	-
invalid	nelze	nelze	nelze	nelze

2.4 Aritmetické výrazy

Aritmetické výrazy jsou tvořené pomocí operátorů, které se dají dělit do tří základních skupin:

1. **unární**, kde je pouze jeden operand (např. proměnná či konstanta),
2. **binární**, kde je operátor zleva i zprava obklopen operandy,
3. **ternární**, který by měl mít okolo sebe tři operandy, ale protože by to šlo těžko, jedná se o operátor složený se dvou znaků, mezi nimiž je prostřední operand(podrobně viz. ternární operátor).

Unární operátory

Běžně známé jsou unární mínus "–" a unární plus "+". Oba se používají stejně, jako jsme zvyklí v matematice. Oba se píší před operand (prefix).
Příklad:

```
var dluh = -21;
var Honza = -(dluh);
```

Speciální unární operátory jsou složené ze dvou stejných znaků a mají význam inkrement "++",
nebo dekrement "--". Oba operátory se dají použít před operandem (prefix), nebo za operandem (suffix).
U prefixu je proměnná nejprve zvětšena o jedničku a pak je tato nová hodnota vrácena jako hodnota výrazu.
U suffixu je vrácena původní hodnota proměnné a ta je pak zvětšena o jedničku.
Příklad:

```
var bluma = 10;
var jablko;
```

jablko = ++bluma; // jablko bude 11, bluma je 11

jablko = bluma++; // jablko bude 11, bluma bude 12

Binární operátory

Rozeznáváme operátory pro:

+ sčítání

- odčítání

* násobení

/ reálné dělení

div celočíselné dělení

% dělení modulo (tj. zbytek po celočíselném dělení např. *10%3* je 1)

Přiřazovací operátory

Přiřazovací operátory jsou určeny k nastavení hodnoty proměnných. Základním přiřazovacím operátorem je "=", jedná se o přiřazení, nikoli o rovnost, která se značí "= =").

Přiřazovací operátor bývá často kombinován s aritmetickými operátory a operátory pro práci s řetězci. Důvodem bývá jednodušší a kratší zápis výrazu:

Tabulka č. 4 – Rozšířené přiřazovací operátory

Rozšířené přiřazovací operátory		
operátor	příklad	ekvivalentní zápis
+=	a += b	a = a + b
-=	a -= b	a = a - b
*=	a *= b	a = a * b
/=	a /= b	a = a / b
div=	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

2.5 Relační operátory

Pomocí relačních operátorů se vytvářejí booleovské výrazy nezbytné v jazykových konstrukcích WMLScriptu, jako jsou např. for, if atd.

Tabulka č. 5 – Relační operátory

Relační operátory			
operátor	*význam*	*operátor*	*význam*
==	rovnost	!=	nerovnost
&&	log. součin se zkráceným vyhodnocováním *	||	log. součet se zkráceným vyhodnocováním*
&	log. součin s úplným vyhodnocováním	|	log. Součet s úplným vyhodnocováním
<	menší	>	větší
<=	menší nebo rovno	>=	větší nebo rovno
!	negace		

*Pozn. Pokud u operátoru && nebo || první vyhodnocovaný výraz je invalid, další vyraz se již nevyhodnocuje a výsledek je invalid.

2.6 Bitové operace

WMLScript má definováno sedm operátorů pro prací s jednotlivými bity čísla. Tyto operace nejsou mnoho rozšířené na mobilních zařízeních, proto není nutné se o nich více rozepisovat.

Tabulka č. 6 – Operátory pro práci s bity

Operátory pro práci s bity	
operátor	*popis*
&	bitový součin - AND
|	bitový součet - OR
^	bitový exklusivní součet - XOR
<<	posun doleva
>>	posun doprava

>>>	neznaménkový posun doprava
~	negace bit po bitu - NOT

2.7 Podmíněné příkazy

Podmíněné příkazy a cykly jsou základem každého procedurálního programovacího jazyka. WMLScript obsahuje stejné podmíněné příkazy jako Java či JavaScript. V následujících odstavcích budou jednotlivě popsány a vysvětleny na příkladech.

Příkaz if a příkaz if-else

Příkaz if je jeden z nejpoužívanějších podmíněných příkazů. Můžeme jej užít v neúplné podmínce – pouze if, nebo v úplné podmínce if – else.

Syntaxe neúplné podmínky je:

```
if (booleovský výraz)
    příkaz;
```

Příklad:

```
if(bluma == 3)
    svestka = 10;       // jestliže bluma je 3,  svestek bude 10
```

Syntaxe úplné podmínky je:

```
if(booleovský výraz)
    Příkaz1;
else                // else se provede pokud není splněna podmínka za if
    Příkaz2;
```

Příklad:

```
if(bluma == 3)
    svestka = 10;    // jestliže bluma je 3,  svestek bude 10
else                 // jinak bude svestek 5
   svestka =  5;
```

Za if (nebo za else) může být jen jeden příkaz. Potřebujeme-li jich uvést více, je třeba použít složený příkaz, kdy se příkazy uzavřou do bloku pomocí závorek "{" a "}" , a tento blok nesmí být ukončen středníkem.

Příklad:
```
if(bluma == 3){
  svestka = 10;     // jestliže bluma je 3,  svestek bude 10
  hruska = 3;       //  a hrusky budou 3
}
```

Ternární operátor (podmíněný výraz)

Operátor se skládá ze dvou znaků "?" a ":" , které jsou obklopeny třemi operandy. Zjednodušeně řečeno, ternární operátor funguje jako if-else. Protože if-else není výraz, ale příkaz, takže nejde použít tam, kde je potřebný výraz. V mnoha případech je to jedno, ale existují místa, kde je použití podmíněného výrazu velmi vhodné. To znamená, že se dá sice obejít, ale za cenu „těžkopádnější" jazykové konstrukce.

Syntaxe podmíněného příkazu:

booleovský_výraz ? výraz1 : výraz2

A má podobný význam jako:

```
if (booleovský_výraz)
  výraz1;
else
  výraz2;
```

Příklad:

var sirka,, vyska= 5; //více proměnných můžeme deklarovat pomocí čárky

sirka = (vyska == 10) : 10 ? 5; // šířka bude 5, protože booleovský výraz je false

2.8 Cykly

Častokrát je zapotřebí provádět jisté akce v WMLScriptu opakovaně, až dosáhneme požadovaného výsledku. Takové opakování kódu se v programování jmenuje iterace nebo **cyklus**. Používáme na to cykly for (doslova pro) nebo while (doslova pokud).

Příkaz while

Tento iterační příkaz používáme jestliže závisí ukončovací podmínka na nějakém příkazu v těle cyklu (předem nedokážeme říci, proběhne-li cyklus jednou nebo stokrát)..

Syntaxe: *while (logický výraz)*
příkaz;

Příklad: *while (bluma < 50)*
bluma++; //cyklus se ukončí pří 49 blumách

Příkaz for

Je to typický příkaz cyklu, který použijeme v případě, že předem známe omezující kriteria, tedy počáteční nastavení, ukončující podmínku a způsob ovlivnění

řídící proměnné. V tomto případě není vhodný cyklus while, protože ten nemá uvedené tři prvky tak přehledně na jednom místě jako for.

Syntaxe: *for (výraz_start; výraz_stop; výraz_iter)*
příkaz;

Příklad:
```
var penize = 0;
for (pocitadlo = 1; pocitadlo < 11; pocitadlo = pocitadlo + 1)
    penize = penize + 100;   // na konci hodnota proměnné peníze bude 1000
```

2.9 Příkazy break a continue

Oba tyto příkazy lze použít ve všech typech cyklů a oba nějakým způsobem mění „normální“ průběh cyklu.

1. break ukončuje nejvnitřnější neuzavřenou smyčku - opouští ihned cyklus.

Příklad:
```
var index = 0, pocitadlo = 0;
while (index <= 10){
    index++;
    if (pocitadlo == 5) break;
    pocitadlo++;
} // počítadlo je 5 a index je 6
```

2. continue skáče na konec nejvnitřnější neuzavřené smyčky (přeskočí zbytek těla cyklu) a tím vynutí další iteraci smyčky - cyklus neopouští.

Příklad:
```
var index = 0, pocitadlo = 0;
while (index <= 10){
    index++;
    if (pocitadlo == 5) continue;
    pocitadlo++;
} // pocitadlo je 5 a index je 10
```

2.10 Pole

Současné verze WMLScript nepodporuje pole. Nicméně standardní String knihovna podporuje funkce pro práci z řetězcem jako s polem.

2.11 Funkce

Funkce se skládá z příkazů a deklarací, které vytváří strukturu daného programu. Proto se kódují spolu a mají společné jméno, což umožní jejich opakované použití, bez opakování kódu jako takového.

Deklarace

Funkce obsahuje jméno, parametry a blok příkazů, který se provede po zavolání funkce Všechny funkce mají následující vlastnosti.

- Úmysl deklarovat funkci naznačíme klíčovým slovem „function“.
- Jméno funkce musí být jedinečné v daném wmls souboru
- Všechny parametry funkce jsou předávány hodnotou.
- Co má funkce vracet je označeno klíčovým slovem return + návratová hodnota.
- Funkce vždy vrací nějakou hodnotu. Není-li uvedena, je vrácen prázdný řetězec.

- Tělo funkce je ohraničeno "{" a "}". Mezi nimi se nachází příkazy a deklarace funkce. Blok je ukončen středníkem.
- Volitelné klíčové slovo extern uvádíme u funkce, která má být přístupná z vnějšku. Takovéto funkce můžeme volat z vnějšku sestavené jednotky, ve které je definována. Zde musí být alespoň jedna funkce definovaná jako přístupná z vnějšku.

Syntaxe: *extern function jméno (parameter1, parameter2) {*

....příkazy a deklarace...

};

Příklad: */* funkce vynasob_cisla ma dva parametry a je přístupná z vnějšku souboru */*

extern function vynasob_cisla (var prvni, var druhy){

*return (první * druhy); // vrátí součet dvou čísel }*

Volání funkce

V jakém případě bude funkce volána záleží na místě deklarace (cílové) funkce. Následující příklady popisují tří způsoby volání funkcí podporované WMLScriptem.

- ***Volání lokální funkce***

  Funkce je volaná uvnitř stejné jednotky.

  Syntaxe: *jméno_funkce(parametr1, ……. parametrX);*

  Příklad:
  ```
  function test1(param){
      return (param * 2);
  };
  function test2(index){
      return test1(index * 3);     // volání lokální funkce test1()
  };
  ```

- ***Volání vnější funkce***

Musí být užito v případě volání funkce deklarované s externího souboru. Definuje mapování mezi externí jednotkou a jménem využívaným uvnitř funkční deklarace. Jméno souboru a „hešovací“ znak (#) je užit jako prefix syntaxe volání funkce.

Pragma **use url** (článek 6.7) musí být uvedena ke stanovení cesty k externímu souboru.

Syntaxe: *jméno_externího_souboru#jméno_funkce(parametry);*

Příklad:
```
use url MeSkripty „http://www.javascript.cz/scripty“; // Pragma
function test2(index){
        return MeScripty#test1(index * 3);
        // volání lokální funkce test1()
};
```

- ***Volání knihovní funkce***

  Funkce je obsažena v nějaké standardní knihovně (WMLSlibs). Před funkcí se uvádí jméno knihovny a tečka.

  Syntaxe: *jméno_knihovny.jméno_funkce(parametry);*

  Příklad:
```
function test2(index){
    return Float.sqrt(Lang.abs(index)+1);
      // volání knihovních funkcí z knihoven Float a Lang
};
```

2.12 Knihovny

Knihovny jsou pojmenované kolekce funkcí, které patří logicky k sobě. Tyto funkce jsou volány pomocí jména knihovny, tečky a názvu dané funkce. Příklad volání knihovních funkcí jsem uvedl v předešlé kapitole.
(více detailů ve wap-194-WmlScript Standard Liberies specification).

V současné době máme 6 základních knihoven:

1. **Lang** - Tato knihovna obsahuje základní funkce WMLScriptu
2. **Float** - Knihovna obsahuje množinu funkcí pro základní aritmetické operace s desetinnými čísly.
3. **String** - Obsahuje funkce pro práci z řetězci.
4. **URL** - Funkce pro práci jak s absolutními tak relativními URL.
5. **WMLBrowser** - Obsahuje funkce, které mohou přistupovat k proměnným v jednotce, která volá tento WMLSkript.
6. **Dialogs** - Knihovna obsahující funkce pro práci s uživatelským rozhraním.

Popis funkcí knihovny Lang

abort()

popis: Ukončí Script a vrátí popis chyby. Tato funkce se užívá ve výjimečných případech, když program může být přerušen chybou v programu.

příklad: *Lang.abort(„chyba:“+ errvalue);*

abs(hodnota)

popis: vrací absolutní hodnotu daného čísla. Číslo může být typu int, nebo typu floating-point

příklad: *var c = -20;*

var e = Lang.abs(c) // e je 20

min(hodnota1, hodnota2)

popis: vrací menší hodnotu ze dvou zadaných hodnot

příklad: *Lang.min(20 , 10); // vrátí 10*

characterSet()

popis: vrací hodnotu typu integer, která určuje charakter skupiny užívané WMLScriptem

příklad: *var a = Lang.characterSet(); // a je 4, (iso-8859-1)*

exit()

popis: ukončí běh skriptu a vrátí zprávu volající aplikaci. Užívá se u funkcí v případech, kdy vykonávání skriptu může být přerušeno

příklad: *Lang.exit(„vse je OK, hodnota je 500“);*

float()

popis: vrací booleanovkou hodnotu true, pokud script podporuje desetinná čísla, jinak false

příklad: *var podporaDesCisla = Lang.float();*

isFloat()

popis: vrací true, pokud hodnota může být konvertována na float. V opačném případě false

příklad: *var a = Lang.isFloat(„-123“); // true*
var b = Lang.isFloat(„3,33“); // true
var c = Lang.isFloat(„ahoj“); // false

isInt()

popis: vrací true, pokud daná hodnota může být konvertována na typ int s využitím funkce parseInt(), jinak false

příklad: *var a = Lang.isInt(„123“); //true*
var b = Lang.isInt(„123,5“); //true
var c = Lang.isInt(„nazdar“); //false

max(hodnota1, hodnota2)

popis: vrátí větší číslo ze dvou zadaných hodnot

příklad: *var vetsi = Lang..max(123. 11); //vráceno bude 123*

maxInt()

popis: vrátí maximální možnou hodnotu vyjádřenou typem int.

příklad: *var max = Lang.maxInt(); // max je 2147483647*

min(hodnota1, hodnota2)

popis: vrátí menší číslo ze dvou zadaných hodnot
příklad: *var mensi = Lang..min(123. 11); //mensi je 11*

minInt(string)
popis: vrátí minimální možnou hodnotu vyjádřenou typem int.
příklad: *var min = Lang.minInt(); // min je -2147483648*

parseFloat(string)
popis: tato funkce vrací desetinné číslo definované jako řetězec
příklad: *var a = Lang.parseFloat("2345.14"); // a = 2345*
var b = Lang.parseFloat(" -4.45e2 Kg"); // b = -4.45e2

parseInt()
popis: funkce vrací Integer definovaný jako řetězec
příklad: *var b = Lang.parseInt("200 m/s"); // b = 200*

random(cislo)
popis: funkce vygeneruje náhodné celé číslo od 0 do zadané hodnoty
příklad: *var a = Lang.random(6.5); // 0 - 6*

seed(cislo)
popis: funkce inicializuje pseudonáhodnou číselnou sekvenci a vrátí prázdný řetězec. Pokud je hodnota 0, nebo kladné celé číslo, potom je daná hodnota užita pro inicializaci a generování náhodných čísel. Pokud je parametr roven 0, nebo je kladné číslo, náhodná čísla se mohou opakovat. V opačném případě se opakovat nebudou.

příklad: *Lang.seed(7);*
var a = Lang.random(6.5); // treba 6
Lang.seed(-2);

var b = Lang.random(6.5); // treba 6

Lang.seed(-2);

var c = Lang.random(6.5); // treba 5. ale neni mozne aby c = b

Popis funkcí knihovny Float

ceil(hodnota)

Popis: funkce zaokrouhluje číslo na celé číslo, které je větší než číslo v parametru funkce

Příklad: *var a = Float.ceil(5.32); // a = 6*

var b = Float.ceil(5.55); // b = 6

var c = Float.ceil(-5.32); // c = -5

floor(číslo)

Popis: funkce zaokrouhluje číslo na celé číslo, které je menší než číslo v parametru funkce

Příklad: *var a = Float.floor(5.32); // a = 5*

var b = Float.floorl(-5.15); // b = -6

int(cislo)

Popis: vrací celé číslo bez desetinných míst

Příklad: *var a = Float.int(5.32); a = 5*

maxFloat()

Popis: vrátí maximální možnou hodnotu vyjádřenou typem float

příklad: *var max = Float.maxFloat(); // max = 3.40282347E+38*

minFloat()

Popis: vrátí minimální možnou hodnotu vyjádřenou typem float

příklad: *var min = Float.minFloat(); // min = 1.17549435-38*

pow(základ, mocnina)

Popis: vrátí mocninu základu

Příklad: *var a = Float.pow(4,2); // a = 16*

var b = Float.pow(-2,3); // b = -8

round(číslo)

Popis: funkce zaokrouhlí na nejbližší celé číslo

Příklad: *var a = Float.round(3.5); // a = 4*

var b = Float.round(3.4); // b = 3

sqrt(číslo)

Popis: funkce vrací odmocninu daného čísla

Příklad: *var a = Float.sqrt(9); // a = 3*

Popis funkcí knihovny String

charAt(string, index)

Popis: funkce vrací řetězec o jednom znaku. Znak v daném řetězci na pozici indexu

Příklad: *var znak = String.charAt(„windows“, 3); // znak = „d“*

compare(String1, String2)

Popis: funkce porovnává dva řetězce na základě číselného kódu každého znaku v řetězci Vrací 0, pokud jsou řetězce identické, 1 pokud první řetězec je větší než druhý. V opačném případě vrací -1.

Příklad: *var a = String.compare("world","world"); // a = 0*

var b = String.compare("I","world"); // b = -1

var c = String.compare("kniha","kuba"); // c = 1

elementAt(*string, index, separator*)

Popis: funkce rozděluje řetězec na podřetězce podle daného separátoru a vrací podřetězec s daným indexem. Pokud je index větší než množství podřetězců, vrací se poslední řetězec.

Příklad *var a = String.elementAt(„Nazdar Karle“, 0, „ „); // a = Nazdar*

var b = String.elementAt(„Nazdar Karle“, 5, „ „); // b = Karle

elements(*string, separator*)

Popis: vrací počet výskytu daného separátoru v řetězci

Příklad: *var a = String.elements(„Nazdar Karle“, „a“); // a = 3*

find(*string, substring*)

Popis: funkce vrací první pozici daného podřetězce v řetězci. Pokud se podřetězec nevyskytuje vrátí funkce hodnotu -1.

Příklad: *var najdi = Sting.find(„okolo“, „kolo“); // najdi = 1*

format(*formatspec, value*)

Popis: funkce konvertuje danou hodnotu na řetězec podle specifikovaných formátovacích pravidel. Formátovací specifikátor má následující úpravu: **% [délka] [.přesnost] typ**

- **délka** je celé kladné číslo určující minimální požadované délky. Na druhou stranu tento argument nikdy nezpůsobí zkrácení dané hodnoty. Pokud délka řetězce je větší než délka udaná argumentem, vypíše se celý řetězec (délka nemusí být udaná vůbec).

- **přesnost** udává nezáporné celé číslo, před kterým je tečka. Užívá se k nastavení přesnosti výstupní hodnoty. Zpracování závisí na zadaném **typu**.
- **d** udává minimální počet tištěných čísel. Pokud počet číslic je menší než udaná hodnota, funkce dorovná daný počet pomocí nul zleva. V případě přesáhnutí dané hodnoty je vypsán celý řetězec. Defaultní hodnota je nastavena na 1.
- **f** specifikuje množství číslic za desetinou čárkou. Defaultní hodnota je nastavena na 6 desetinných míst. Číslo se zaokrouhluje na odpovídající hodnotu. Pokud je hodnota 0, nebo je uvedená pouze tečka, desetinná čárka se nevypisuje.
- **s** vyjadřuje maximální počet znaků, které mohou být tištěny, implicitně je nastavena na všechny.
- **typ** je jediný argument, který musí být specifikován. Typ určuje, zda daná hodnota bude interpretována jako integer, floating-point, nebo string. Pokud hodnota argumentu je odlišná od typu argumentu, automaticky se konvertuje na daný typ. Podporované typy jsou **d** pro integer, **f** - floating-point a **s** pro řetězec. Řetězec je tisknut do daného znaku. Pokud je délka větší jak zadaná hodnota, potom se nadbytečné znaky ignorují.

Příklad: *var b = String.format("%4.3d", 32); // b = " 032"*
var d = String.format("%3f", 10.1234); // d = "10.123"
var e = String.format("%2.2f", 2.3); // e = "2.30"

insertAt(*string, substring, index, separator*)

Popis: užívá se pro vložení řetězce do jiného řetězce. Parametry funkce jsou řetězec a podřetězec, separator určující členění řetězce a index před který se připojí podřetězec.

Příklad**:** *var a = String.insertAt("Vítej Karle","u nás",1," ");//Vítej u nás Karle*

isEmpty(*string*)

Popis: funkce vrací booleanovskou hodnotu true pokud jde o prázdný řetězec jinak vrací false

Příklad: *var a = String.isEmpty(""); // a = true*

var b = String.isEmpty("Hello world"); // b = false

var c = String.isEmpty(23.4); // c = false

length(*string*)

Popis: vrací délku řetězce

Příklad: *var a = String.length(""); // a = 0*

var b = String.length("Hello world"); // a = 11

removeAt(*string, index, separator*)

Popis: funkce dělí řetězec na dané podřetězce podle separátoru a poté odsraní podřetězec s daným indexem. Pokud je index větší než počet podřetězců, vymaže se poslední podřetězec.

Příklad: *var a = String.removeAt("Dobry den!",0," "); // a = „Dobry“*

var b = String.removeAt("Dobry den!",0,"o"); // b = „ bry den“

replace(*string, oldstring, newstring*)

Popis: funkce zamění, nahradí část řetězce novým řetězcem a vrátí výsledek

Příklad: *var a = String.replace("Lukas","Lu","To"); // a = „Tomas“*

replaceAt(*string, substring, index, separator*)

Popis: funkce rozděluje řetězec na dané podřetězce podle separátoru a poté nahradí indexovaný podřetězec novým řetězcem

Příklad: *var a = String.replaceAt("Miluju tebe!","Maluju",0," "); // a = „Maluju tebe“*

squeeze(*string*)

Popis: redukuje více bílých znaků za sebou na jeden a vrátí daný řetězec

Příklad: *var a = String.squeeze("Velky Rum \n\t"); // a = „Velky Rum „*

subString(*string, start, length*)

Popis: vrací podřetězec specifikovaný indexem a délkou podřetězce

Příklad: *var a = String.subString("Petra",4,1); // a = Petr*

toString(*value*)

Popis: vrací hodnotu typu string

Příklad: *var a = String.toString(66); // a = „66“*

trim(*string*)

Popis: funkce odstraní bílé znaky před a za řetězcem

Příklad: *var a = String.trim(" atd. "); // a = „atd.“*

Popis funkcí knihovny URL

isValid(URL)

Popis: vrací booleanovskou hodnotu true pokud je zapsána správně syntaxe URL, jinak vrací false. Funkce podporuje zápis absolutní a relativní adresy.

Příklad: *var a = URL.isValid("htt//www.seznam.cz"); // a = false*

var b = URL.isValid("http://www.seznam.cz!!"); // b = true

getScheme(URL)

Popis: vrací protokol dané adresy

Příklad: *var a = URL.getScheme("http://www.w3schools.com");// a = „http“*

var b = URL.getScheme("www.w3schools.com");// b = „ “

var c = URL.getScheme("ftp://www.w3schools.com"); // c = „ftp“

getHost(URL)

Popis: vrací adresu specifikovanou jako URL parametr. Pokud adresa není specifikována, vrátí funkce prázdný řetězec

Příklad: *var a = URL.getHost("http://www.seznam.cz/index.htm");//a = www.seznam.cz*

var b = URL.getHost("/wml/funce");// b = „ “

getPort(URL)

Popis: vrací číslo portu udaného v URL adrese. Není-li číslo portu uvedeno, vrátí se prázdný řetězec

Příklad: *var a = URL.getPort("http://w3schools.com:80");//a = „80“*

var b = URL.getPort("http://w3schools.com");//b = „ “

getPath(URL)

Popis: vrací cestu specifikovanou URL adresou. Není-li uvedena, vrací prázdný řetězec.

Příklad: *var a = URL.getPath("http://w3schools.com/wml/tips.htm"); // a = "/wml/tips.htm"*

var b = URL.getPath("http://w3schools.com");// b = „ “

getParameters(URL)

Popis: funkce vrací parametr obsažen v poslední části cesty URL parametru

Příklad: *var a = URL.getParameters("http://w3schools.com/wml;tip");// a = "tip"*

var b = URL.getParameters("http://w3schools.com/wml");// b = „ "

getQuery(URL)

Popis: vrací ověřovací část ve specifikovaném URL

Příklad: *var a=URL.getQuery("http://w3s.com/go.asp?name=bill");// a = „bill“*

var b=URL.getQuery("http://w3s.com");// b = „ “

getFragment(URL)

Popis: funkce vrací fragment dané URL

Příklad: *var a = URL.getFragment("http://w3schools.com/wml#read");//a = „read“*

var b = URL.getFragment("http://w3schools.com/wml");// b = „ “

getBase()

Popis: vrací absolutní URL (bez fragmentu) současného scriptu

Příklad: *var a = URL.getBase(); // vrátí adresu kde je script uložen*

getRefer()

Popis: vrací nejmenší relativní URL daného souboru

Příklad: *var base = URL.getBase();*

// base = "http://www.host.com/current.scr"

var referer = URL.getReferer();

// referer = "app.wml#card2"

resolve(vychoziURL, vlozeneURL)

Popis: funkce navrací absolutní URL složené z výchozího URL a vloženého URL. Nejdříve je ověřeno výchozí URL (pokud se jedná o prázdný řetězec, je doplněno za výchozí URL lomítko). Jestliže je zapsána chybná syntaxe URL, je vracena hodnota invalid.

Příklad: *var a = URL.resolve("http://wap.cz/","okno.vcf");*

// a = "http://wap.cz/okno.vcf"

var b = URL.resolve("http://wap.cz","tip");

// b = "http://wap.cz/tip"

var c = URL.resolve("http://wap.cz","/tip");

// c = "http://wap.cz/tip"

escapeString(URL)

Popis: nahradí speciální znaky v URL hexadecimálními escape sekvencemi a vrátí výsledek.

Příklad: *var a = URL.escapeString("http://jahoda.cz/wml/");*

// a = "http%3a%2f%2fjahoda.cz%2fwml%2f"

unescapeString(*string*)

Popis: nahradí escape sekvence v URL odpovídajícími znaky a vrátí výsledek

Příklad: *var a = "http%3a%2f%2fjahoda.cz%2fwml%2f"*

// a = URL.escapeString("http://jahoda.cz/wml/

loadString(URL, danyTyp)

Popis: vrací obsah vyjádřený absolutním URL a daným typem. Nesmí být udáno více datových typů. Typ musí být text (text/další_typ) ,ale další typ může býti cokoliv.

Příklad: var myUrl = "http://www.host.com/vcards/myaddr.vcf";

myCard = URL.loadString(myUrl, "text/x-vcard");

Popis funkcí knihovny WMLBrowser

GetVar(jmeno)

Popis: vrací hodnotu proměnné, která je uvedená jako parametr v souvislosti s prohlížečem.

Příklad: *var a = WMLBrowser.getVar("mena");*

// může vrátit například a = "Kc"

SetVar(jmeno,hodnota)

Popis: funkce nastaví hodnotu proměnné, které se zobrazují na stránce. Jméno proměnné je zadáno jako první parametr. Vrátí true v případě úspěšného nastavení nové hodnoty, v opačném případě navrátí false.

Příklad: *var a = WMLBrowser.setVar("unor", 28);*

// v pripade uspechu a = true

go(URL)

Popis: pomocí této funkce přejdeme na novou kartu specifikovanou danou adresou a vrátí prázdný řetězec

Příklad: *var hlavni = "http://www.aaa.cz/auto/skoda.dck#start";*

var a = WMLBrowser.go(hlavni); // var a = ""

prev()

Popis: funkce přejde na předcházející kartu a vrátí prázdný řetězec. Tato funkce má stejný význam jako element <prev> ve WML.

Příklad: *var a = WMLBrowser.prev(); // var a = "";*

newContext()

Popis: funkce vymaže všechny proměnné v souvislosti s WML a odstraní historii vyjma současné karty. Vrací prázdný řetězec.

Příklad: *var a = WMLBrowser.newContext(); // var a = "";*

getCurrentCard()

Popis: vrací nejmenší relativní URL současné karty

Příklad: *var a = WMLBrowser.getCurrentCard();*

// a = "deck#druhy"

refresh()

Popis: tato funkce překreslí kartu a zaktualizuje zobrazení proměnných. Vrací prázdný řetězec, pokud aktualizace byla provedena úspěšně. V opačném případě bude vrácen neprázdný řetězec.

Příklad: *var a = WMLBrowser.setVar("day",11);*

var b = WMLBrowser.refresh();//a = true, b = ""

Popis funkcí knihovny Dialogs

prompt (info, default_hodnota)

Popis: funkce zobrazí informaci na displeji a čeká na odpověď uživatele. Druhý parametr je implicitní hodnota, která bude vrácena pokud uživatel nezapíše žádnou hodnotu. Jinak bude vrácena hodnota zadaná uživatelem.

Příklad: *var a = Dialogs.prompt("Zadejte cislo:","0");*

//a = "5" pokud uzivatel zada cislo 5

a = "0" pokud uzivatel nic nezada

confirm(info, ok, cancel)

Popis: funkce zobrazí danou zprávu na displeji a dvě možnosti odpovědi. Pokud zvolí odpověď ok vrátí true, v druhém případě je vrácena hodnota false.

Příklad: *var a = Dialogs.confirm("Zmenit?","Yes","No");*

//a = true (jestliže bylo zvoleno "Yes") a = false (zvoleno "No")

alert(info)

Popis: zobrazí danou zprávu a čeká na potvrzení od uživatele. Funkce vrací prázdný řetězec.

Příklad: *var a = Dialogs.alert("Musite zadat cislo!"); // a = ""*

2.13 Direktiva (pragma)

Direktiva se zapisují před funkce a jsou označeny klíčovým slovem use a uvedením dalších specifických parametrů. Využívají se ve třech případech. Jedná se o přístup k funkcím v externím zdrojovém souboru, kontrolu přístupu a o meta informace.

Externí kompilační jednotka

Určité aplikace mohou využívat oddělené skripty, které již byli individuálně a nezávisle zkompilovány. Jednotka může tyto funkce volat pomocí direktiv (ukazatele na jiné zdroje, které budou užívány). Soubor může přistoupit na adresu specifikovanou názvem souboru a jeho URL pragma musí být užita, pokud voláme externí funkce.

Zápisem use URL direktiva specifikujeme umístění (URL) externího zdroje s WMLScriptem a přidělíme mu místní jméno. Toto jméno může být používáno uvnitř deklarace funkce k vytvoření externího volání funkce.

Prohlédněme si následující příklad:

```
use url DocJS "http://www.docjavascript.com/mylibs/mywmlscript"
function divide(nom, denom) {
  if (denom == 0) return invalid;
  return DocJS#mydiv(nom, denom);
}
```

Chování tohoto příkladu je následovné:

- **Pragma stanovuje url k dalšímu WMLScriptu**
  Use url direktiva specifikuje umístění externí zkompilovaného souboru, v našem případě je cesta k souboru „http://www.docjavascript.com/mylibs/mywmlscript“. Jestiže není udána přípona souboru, program bude hledat příponu .wmlsc .
- **Funkce zavolá jednotku s tímto url**
  Jakmile je soubor naleznut, volání funkce “DocJS#mydiv()“ spustí nahrávání externího souboru http://www.docjavascript.com/mylibs/mywmlscript.wmlsc. Direktiva má své vlastní jméno (DocJS), které musí být jedinečné uvnitř zdrojového souboru..
- **Obsah je prověřen a daná funkce se vykoná**

Kontrola přístupu (access control pragma)

WMLScript může chránit vlastní obsah souboru užitím direktivy kontroly přístupu. Kontrola přístupu je provedena před přístupem k externím souborům. Jedna jednotka může obsahovat pouze jednu direktivu pro kontrolu přístupu. Ta specifikuje z jakého umístění můžeme volat externí funkce uložené v souboru s kontrolou přístupu. Syntaxe zápisu může být zapsána jedním z následujících způsobů:

use access domain
use access domain ;
use access path ;
use access domain path ;

URL odkazujících se kompilačních jednotek se musí shodovat ve specifikaci kontroly přístupu. URL zahrnuje jméno domény a cestu. Jakmile se domény shodují, je vyhodnocena uvedená cesta, kde veškeré prvky sub-domén se musí shodovat.
Příklad: *use access domain "webreference.com" path "/js";*

Následujícím URL je dovoleno volat externí funkce u jednotky s kontrolou přístupu.
http://webreference.com/js/tips.cgi
https://www.webreference.com/js/categories.wmlsc
http://www.webreference.com/js/tools/newtool.cgi?x=123<y=456

Tyto URL nemají možnost přístupu k externím funkcím.
http://www.microsoft.com/javascript
http://www.webreference.com/dhtml/mytool.cgi

Jak jste již mohli poznat, přístupová cesta se shoduje v části cesty z leva doprava a musí se shodovat v celé uvedené části cesty. Pokud v souboru nejsou uvedeny žádná přístupová práva, je soubor automaticky veřejně přístupný.

Meta informace (Meta-information)

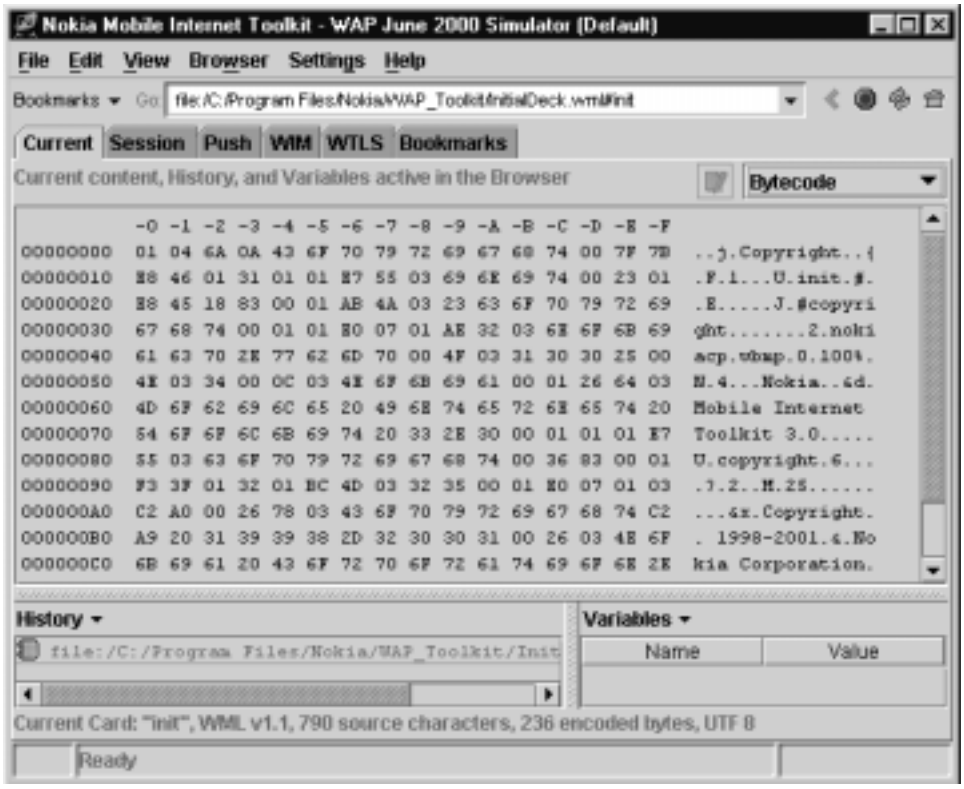
Pomocí meta dat předáváme informace pro prohlížeče, servery a jiné aplikace o daném souboru. Je to nepovinné pro všechny soubory. Meta data mohou obsahovat tři atributy: **vlastní jméno, obsah, volitelné schéma**. Příklad jednotlivých použití meta-informací :

use meta name "Created" "22-May-00";
use meta http equiv "Keywords" "Script, Language";
use meta user agent "Type" "Test";

2.14 Binární soubory

Kód WML a WMLScriptu musí být před svým používáním zkompilovány do binární podoby (tzv. byte-code) na straně serveru a potom poslány do prohlížeče klienta. Výhoda binárních souborů spočívá ve snadném převedení přímo do strojového kódu libovolného procesoru (podobně jako tomu je v jazyce Java). Soubor zkompilujeme jednou a poté je spustitelný na jakémkoliv WML prohlížeči . Binární soubor zabírá méně než jednu polovinu psaného kódu. Zobrazit binární kód můžeme v různých vývojových prostředí (nejlepší podle mého názoru je vývojové prostředí Nokia WAP Toolkit 3 z kterého je následující obrázek).

Obrázek č. 1 – Binární kód



Každý byt je reprezentován dvěma hexadecimálnímy znaky a je ohraničen bílými znaky. Levý sloupec udává číslo řádku od začátku souboru. Poslední sloupec obsahující ASCII znaky může být použit pro rekonstrukci originálního WML kódu.

3. WTAI (Wireless Telephony Application Interface)

Pomocí WTAI funkcí (Wireless Telephony Application Interface) můžeme ovládat rozhraní mobilního zařízení jako je například vytáčení hovorů, odesílání SMS a práce s adresářem na SIM kartě. Není proto žádný problém napsat aplikaci, která podle dotazu uživatele vyhledá v databázi telefonní číslo a po potvrzení uživatelem je číslo automaticky vytočeno. WTAI tak zajišťuje další rozvoj dynamických wapových aplikací. Funkce WTAI můžeme začlenit přímo do WML, nebo do WMLScriptu. WTAI tedy uživateli nabízí další možnosti pro ovládání služeb souvisejících s telefonováním, tedy s tím, na co byly (a snad ještě i jsou) telefony určeny.

Tady je jejich jen velmi zkrácený výčet:

- operace s příchozím hovorem: přijmutí, přesměrování (do schránky, popř. předání na jiné telefonní číslo)
- uskutečnění nového hovoru
- práce s SMS zprávami
- práce s telefonním seznamem na SIM kartě
- ovládání hlasové schránky: zjištění seznamu zpráv, výběr zpráv pro přehrání, forwardování zpráv
- práce s USSD příkazy
- vysílání DTMF tónů
- jiné (dnes již běžné) funkce: konferenční hovor, přidržení a přesměrování hovorů

3.1 Knihovny WTAI

Základním kamenem pro používání WTAI funkcí jsou knihovny, které jsou obsahově rozděleny dle základních nabízených služeb. Knihovny WTAI funkcí jsou přístupné z WMLScriptu použitím knihoven skriptovacích funkcí. Některé obecné funkce, zajišťující rozhraní mezi mobilem a sítí, jsou také přístupné přímo z WML pomocí URI.

Příklad použití WTAI pomocí speciálních funkcí WMLScriptu:

WTAPublic.makeCall("0609111333");

//funkce vytočí číslo udané jako parametr

Příklad použití odkazů, které specifikují URL adresy WTAI funkcí:

Zavolej mi do klubu!

// docílení stejného výsledku jako v předcházejícím scriptu pomocí URI funkce.

Některé funkce se mohou ovládat pouze pomocí WMLScriptu:

WTAPhoneBook.write("", "0609111333", "Dlouha Berta");

// přidá se záznam do seznamu na SIM kartu

3.2 Schéma funkcí URI

Přístup k nějaké funkci z WTAI knihovny ve WML dokumentu se provádí protřednictvím URI volání. Jedná se o předdefinované odkazy na dané knihovny WTAI funkcí. Příklad takovéto předdefinované knihovny je "WTAPublic", specifikující veřejné WTAI funkce. WTAI funkce jsou přejmenovávány pomocí URL.

Syntaxe: *wtai://<knihovna>/<funkce> (; <parametr>)* [! <návratové proměnné>]*

Vysvětlivky k daným položkám:

- *<knihovna>* Název funkce, který identifikuje typ funkce, např. WTAPublic používá název knihovny "wp".
- *<funkce>* Identifikátor funkce uvnitř specifikované knihovny. Příkladem je "mc" pro funkci "makeCall" (sestavení nového hovoru), který je součástí knihovny "WTAPublic".
- *<parametr>* Žádný, nebo více parametrů, které mohou byt poslány funkci. Oddělovač mezi následujícími parametry musí být středník ";".
- *<návratová hodnota>* Začátek sekce návratových dat je indikován vykřičníkem "!". Výsledek volitelného názvu proměnné, který

bude nastaven v kontextu uživatelské aplikace WTA jako výsledek volané funkce.

Knihovny jsou především rozděleny na obecně zaměřené (nezávislé na technologii) a specifické pro daný typ sítě (např. síť GSM operátora). Následující tabulka ukazuje stručný přehled těchto knihoven, které jsou jednotlivě rozepsány v dalších podkapitolách :

Tabulka č. 7 – Obecné knihovny WTAI

Obecné knihovny WTAI	
Knihovna funkcí	Stručný popis knihovny
WTAPublic	Knihovna veřejných funkcí (funkcí na nejobecnější úrovni).
WTAWoiceCall	Kontrola hlasového volání. Držení nastaveni voláni a kontrola zařizení v průběhu trvajícího volání.
WTANetText	Sitova textova knihovna. Posila a prijima sitove texty (zpravy).
WTAPhoneBook	Knihovna telefonniho seznamu (adresar). Ridi vstupy do telefonniho seznamu zarizeni.
WTACallLogs	Knihovna zaznamu o volani. Pouziva se pro pristup k ruznym druhum zaznamu o volani v zarizeni
WTAMisc	Zachytavani ruznorodych rysu. Prikladem je logicka indikace.

Knihovna WTAIPublic (Veřejná knihovna VTAI)

V této knihovně můžete najít základní funkce:

- ***Funkce WMLScriptu:***

WTAIPublic.makeCall(číslo)

Popis: Tato funkce vytočí dané telefonní číslo, které je uvedeno jako parametr. Tato funkce navrátí prázdný řetězec pokud spojení bylo úspěšně provedeno. V opačném případě se vrátí záporná integer reprezentující chybové hlášení, nebo typ invalid pokud funkce selže. Funkce může vrátit následující hodnoty:

- -105 = volaná linka je obsazena

- -106 = selhalo síťové spojení
- -107 = na volané lince není odezva
- -1 = nespecifikovaná chyba

Příklad: *var spojeni = WTAIPublic.makeCall("0607656977");*
//proměnná spojení bude nabývat hodnoty –105, pokud linka bude obsazena

WTAIPublic.sendDTMF(dtmf)

Popis: Poslání DTMF tónů přes hlasové volání. Dtmf parametr specifikuje DTMF sekvenci, která bude poslána (musí to být řetězec čísel). Častěji se užívá WTAIPublic.makeCall , nebo
wtai://wp/mc funkce.
Návratové hodnoty:

- "" = úspěch (vrací prázdný řetězec).
- -108 = nenavázáno spojení
- -1 = nespecifikovaná chyba

Příklad: *var spojeni = WTAIPublic.sendDTMF("0607*656977");*

WTAIPublic.AddPBEntry(číslo,jméno)

Popis: Přidávání záznamu do telefonního seznamu pomocí této funkce se dvěma atributy pro telefonní číslo a jméno účastníka. Jméno může být i prázdný řetězec.

Vrací hodnoty:

- "" = úspěch (vrací prázdný řetězec).
- -100 = jméno je nepřijatelné, nebo příliš dlouhé
- -101 = neplatné telefonní číslo uvedené v parametru
- -102 = příliš dlouhé telefonní číslo
- -103 = nelze zapsat do telefonního seznamu
- -104 = telefonní seznam je plný

- -1 = nespecifikovaná chyba

Příklad: *var spojeni = WTAIPublic.addPBEntry(“0606652973“, “Servac“);*

- ***Funkce URI***

wtai://wp/mc;**číslo**

Popis: Funkce pro vytočení čísla stejně jako tomu bylo u funkce makeCall(číslo).

Návratové hodnoty jsou identické jako u WTAIPublic.makeCall a ještě je přidána hodnota

-200 pro chybné volání.

příklad: *<go href= “wtai://wp/mc;0606652973“/>*

wtai://wp/sd;**dtmf**

Popis: Vyslání DTMF tónu. Návratové hodnoty jsou stejné jako u WTAIPublic.sendDTMF rozšířeno o hodnotu –200 pro chybné volání.

Příklad: *<go href=“wtai://wp/sd;0606*652973“/>*

wtai://wp/ap;číslo;jméno

Popis: Přidávání záznamu do telefonního seznamu. Jméno a číslo se musí zobrazit uživateli na displeji pokud je povolen zápis. Návratové hodnoty jsou stejné jako u funkce WAIPublic.addPBEntry + rozšíření o hodnotu -200 = chybné volání.

Příklad: *<go href=“wtai://wp/ap;0606652973;Servac“/>*

Knihovna WTAWoiceCall

Knihovna pro nakládání s hovory

- ***Události WTA***

Tyto události se vztahují k hlasovému volání. Všechny parametry událostí WTA jsou vyjádřeny jako řetězce.

- **wtaev-cc/ic -** Indikuje příchozí volání. Hovor se může přijmout pomocí funkce WTAVoiceCall.accept, nebo odmítnout funkcí WTAVoiceCall.release.
- **wtaev-cc/cl** - Indikuje zrušení daného volání. Vrácený výsledný parametr ukazuje jak bylo volání zrušeno. Musí se jednat o jednu z následujících hodnot:
    - "0" = normální přerušení hovoru
    - "1" = o ukončení nejsou žádné informace
    - "2" = příčina v síti
    - "3" = ukončení pro výpadek signálu
    - "4" = volaná linka je obsazena
    - "5" = není dostupná síť
    - "6" = stanice neodpovídá
- **wtaev-cc/co** - Indikuje příchozí a odchozí volání.
- **wtaev-cc/oc** - Indikace spojení voláni.
- **wtaev-cc/cc** - Vyzvánění hovoru.
- **wtaev-cc/dtmf** - Vyslání DTMF tónu.

- ***Funkce WMLScriptu***

WTAVoiceCall.setup(číslo, mód)

Popis: Sestavení nového hovoru, kdy jako parametr udáváme číslo a mód, který je typu boolean. Pokud chceme uchovat hovor ještě po přerušení, bude mód nastaven na true, na false nastavíme v případě, že nechceme hovor držet v paměti pří ukončení.

Příklad: *var hovor = WTAVoiceCall.setup("0606652973", true);*

WTAVoiceCall.accept(volání, mód)

Popis: Funkce pro přijmutí příchozího hovoru (prakticky se jedná o zvednutí sluchátka). První parametr určuje volající číslo, které čeká na odpověď. Druhým parametrem je mód pro uchování hovoru v paměti. Pokud funkce selhala vrátí invalid, jinak vrací prázdný řetězec.

Příklad: *var hovor = WTAVoiceCall.accept(handle, false);*

WTAVoiceCall.release(volání)

Popis: Zrušení hovoru, který je zapsán jako parametr. Vrací prázdný řetězec, nebo invalid.

Příklad: *var hovor = WTAVoiceCall.release(handle);*

WTAVoiceCall.sendDTMF(volání, dtmf)

Popis: Vysláni DTMF tónu.

Příklad: *var hovor = WTAVoiceCall.sendDTMF(handle,"555*777666");*

WTAVoiceCall.callStatus(volání, pole)

Popis: Každé WTA zařízení poskytuje určité informace o volání. Každá položka obsahuje jméno a hodnotu. Hodnotu pole dostaneme při zadání jména pole. Následující položky jsou dostupné pro každé hlasové volání:

“number“ obsahuje telefonní číslo

“status“ Integer ukazuje současný stav volání

“duration“ vrací délku hovoru (není moc přesný).

Příklad: *var hovor = WTAVoiceCall.callStatus(handle,“name“);*

WTAVoiceCalll.ist(hovor1)

Popis: Zjištění informací o předchozím volání. Tato funkce je volaná opakovaně ke zjištění všech předešlých volání. Parametr je typu boolean. Pokud je nastaven na true, vrátí nejstarší hovor, false vrací další hovor v pořadí, nebo je vracena false pokud již žádný není (konec seznamu).

Příklad: *var old = WTAVoiceCall.CallList(true); // nejstarší hovor #1*

var a = WTAVoiceCall.CallList(false); // další hovor v pořadí #2

var b = WTAVoiceCall.CallList(false); // #3

var c = WTAVoiceCall.CallList(true) ; // #1

var d = WTAVoiceCall.CallList(false); // #2

Knihovna WTANetText

- ***Události WTA***

Tyto události uvádějí vztah k síťovým zprávám. Všechny parametry události WTA jsou vyjádřeny jako řetězce.

- **wtaev-nt/st** indikuje poslané zprávy
- **wtaev-nt/it** indikuje příchozí síťovou zprávu

- ***Funkce WMLScriptu***

WTANetText.send(adresa, text)

Popis: Poslání textové zprávy na adresu. V případě selhání funkce je vrácen typ invalid, nebo je vygenerováno chybové hlášení:

- -100 = příliš dlouhá zpráva
- -1 = neidentifikovaná chyba

Příklad: *var posli = WTANetText.send(0606652973, "Ahoj");*

WTANetText.list(zpráva1, typZprávy)

Popis: Výpis příchozích zpráv. Funkce je volaná opakovaně ke zjištění všech zpráv. První parametr je typu boolean. Pokud je nastaven na true je vyhledána poslední zpráva, pomocí false je určena další zpráva v seznamu. Typ invalid je vrácen v případě, že již není žádná další zpráva.

Parametr udávající typ určuje druh zprávy, který je zadán jedním z těchto číslel:

- 0 = zahrnuje veškeré čtené, nečtené a neposlané zprávy
- 1 = zahrnuje pouze nečtené zprávy
- 2 = pouze čtené zprávy
- 3 = zahrnuje neposlané zprávy

Pokud je ale předchozí parametr nastaven na false je typ ignorován.

Příklad: *var cti* = WTANetText.list(true, 1)*; // nejstarší nečtená zpráva - #1*

var a = WTANetText.list(false, 1)*; // #2*

var b = WTANetText.list(true, 2)*; // nejstarší čtená zpráva - #3*

var c = WTANetText.list(false, 3)*; //typ je ignorován - #4*

WTANetText.remove(zpráva1)

Popis: Vymazání příchozí, nebo odchozí zprávy ze zařízení. Parametr specifikuje, jaká zpráva má být vymazána. Při úspěšném vymazání vrací prázdný řetězec, v opačném případě chybový kód (-101 = zpráva nemohla být vymazána, -1 = nespecifikovaná chyba).

Příklad: *var kos = WTANetText.remove(hendle);*

WTANetText.getFieldValue(zpráva1, pole)

Popis: První parametr identifikuje danou zprávu a druhý požadovanou informaci .WTA zařízení poskytuje určité informace o zprávě. Každá položka obsahuje jméno a hodnotu. Hodnotu pole dostaneme při zadání jména pole.

Následující položky jsou dostupné pro všechny zprávy:

- **"text"** = string obsahující samotnou zprávu
- **"tstamp"** = string obsahující časový údaj o příchozí zprávě.
- **"address"** = string s adresou odesílatele u příchozí zprávy a adresou příjemce při

  odeslání zprávy

- **“read“** = typ boolean, který ukazuje jestli již byla zpráva označena jako přečtená

  (jestiže je hodnota true), nebo nepřečtená (hodnota false)

Příklad: *var m_text = WTANetText.getFieldValue(hadndle, “address“);*

WTANetText.remove(zpráva1)

Popis: Vymazání příchozí, nebo odchozí zprávy ze zařízení. Parametr specifikuje, jaká zpráva má být vymazána. Při úspěšném vymazání vrací prázdný řetězec, v opačném případě chybový kód (-101 = zpráva nemohla být vymazána, -1 = nespecifikovaná chyba).

Příklad: *var kos = WTANetText.remove(hendle);*

WTANetText.markAsRead(zpráva1)

Popis: Označí zprávu určenou v parametru za přečtenou a vrátí prázdný řetězec.

Příklad: *var aaa = WTANetText.markAsRead(hendle);*

Knihovna WTAPhonebook (telefonní seznam)

- ***Události WTA***

S touto knihovnou nejsou asociovány žádné události. Žádné události nejsou generovány jako přímý, nebo nepřímý výsledek voláni funkce z této knihovny.

- ***Funkce WMLScriptu***

WTAPhoneBook.write(index, číslo, jméno)

Popis: Funkce přidá záznam do seznamu, nebo přepíše existující zápis. Index specifikuje dané místo v telefonním seznamu. Pokud je index roven nule, číslo je uloženo na právě volnou pozici. Dále se uvádí číslo a s ním asociované jméno. V případě úspěchu funkce vrací index záznamu. Při neúspěchu je vrácen chybový kód (stejný jako u WTAIPublic.AddPBEntry).

Příklad: *var uloz = WTAPhoneBook.write(index, číslo, jméno);*

WTAPhoneBook.search(pole, hodnota)

Popis: Vrací index položky splňující zadaná kritéria. Pokud je seznam prázdný, není vrácen žádný index. Parametr pole určuje jaká oblast v seznamu se bude prozkoumávat. Hodnota udává hledaný řetězec (nesmí obsahovat čárky, nerozlišují se malá a velká písmena).

Příklad: *var index = WTAPhoneBook.search("name", "pave");*

WTAPhoneBook.remove(index)

Popis: Vymaže záznam určený indexem v adresáři a vrátí prázdný řetězec. Případně vygene-ruje chybový kód (-105 = nelze smazat, nebo –1 = neznámá chyba).

Příklad: *var x = WTAPhoneBook.remove(index);*

WTAPhoneBook.getFieldValue(index, pole)

Popis: Vrátí hodnotu pole ze specifikovaného záznamu. Index uvádí místo v telefonním seznamu. Tato funkce vrací hodnotu, která záleží na stanoveném parametru pole, který musí obsahovat každý záznam. Jedná se o "number" (String obsahující telefonní číslo) a o "name"(String se jménem).

Příklad: *var x = WTAPhoneBook.getFieldValue(index, pole);*

WTAPhoneBook.change(index, pole, nováHodnota)

Popis: Funkce určená pro editaci záznamů záznamu. Záznam specifikovaný indexem a polem, které se bude editovat je změněn pomocí parametru s novou hodnotou.
V případě úspěchu se vrací prázdný řetězec a při neúspěchu chybový kód:

- -100 = parametr pole "name" obsahuje neakceptovatelné znaky, nebo je příliš dlouhý
- -101 = Nová hodnota vkládaná do pole "number" není telefonní číslo.
- -102 = Nová hodnota "number" je příliš dlouhá.
- -103 = Záznam nemohl být změněn.
- -104 = Záznamník je plný.
- -106 = Byl uveden nepodporovaný parametr pole.
- -107 = Nebyl zadán správný parametr a nesprávná nová hodnota.
- -1 = Byla nalezena nespecifikovaná chyba.

Příklad: *var x = WTAPhoneBook.change(index, pole, nováHodnota);*

Knihovna WTACallLog (záznamy o hovorech)

Knihovna pro práci se záznamy o hovorech.

- ***Události WTA***

S touto knihovnou nejsou asociovány žádné události.

- ***Funkce WMLScriptu***

WTACallLog.dialled(záznam1)

Popis: Zjištění volaných čísel. Funkce se volá opakovaně pro zjištění všech záznamů. Pokud je parametr nastaven na true, je vrácen poslední volaný záznam. Nastavení na false zjistí další záznam v pořadí.

Příklad: *var a = WTACallLog.dialled(true); // přečte záznam #1*

var b = WTACallLog.dialled(true); // přečte záznam #2

WTACallLog.missed(záznam1)

Popis: Zjištění zmeškaných hovorů pomocí stejných pravidel jako v předešlé funkci.

Příklad: *var zmeskane = WTACallLog.missed(true); // přečte zmeškaný záznam #1*

WTACallLog.received(záznam1)

Popis: Zjištění přijatých hovorů.

Příklad: *var zmeskane = WTACallLog.received (true); // přečte přijatý záznam #1*

WTACallLog.getFieldValue(hovor, pole)

Popis: Vrátí hodnotu pole ze specifikovaného hovoru. První parametr určuje identifikační číslo hovoru a parametr pole druh hledaného záznamu. Funkce vrací hodnotu, která záleží na stanoveném parametru pole, který musí obsahovat každý záznam. Jedná se o “number“ (String obsahující telefonní číslo) a o “tstamp“(String obsahující datum a čas).

Příklad: *var tcislo = WTACallLog.getFieldValue(hovor, “number“);*

Knihovna WTAIMisc (ostatni)

Tyto funkce mají své specifické využití pro tvorbu aplikací. Proto se omezuji pouze na jejich výčet a stručnou charakteristiku. V případě zájmu jsou dostupné ve formě PDF dokumentu na stránkách “www. wapforum.org“.

- **WTAMisc.setIndicator** - vypnutí/zapnutí indikace
- **WTAMisc.endContext** - ukončení kontextu klienta
- **WTAMisc.getProtection** - získáni informací o ochraně kontextu před přerušením
- **WTAMisc.setProtection** - nastavení ochrany kontextu před přerušením

4. Skriptování pomocí PHP

Zatím jsem se zabýval hlavně WMLScriptem, který „běží“ na straně klienta. V případě WAPu tedy na telefonu, nebo v jeho emulátoru. Klient má možnost zobrazit zdrojový tvar těchto skriptů.

Jsou ovšem případy, kdy nám toto skriptování stačit nebude. Jedná se hlavě o přístup k různým databázím, nebo posílání emailových zpráv. Tyto aplikace „neběží“ přímo v prohlížeči, ale jsou umístěny na straně serveru (jedná se o tzv. ServerSide Script). Nejprve stránku vygenerují a teprve tu posílají klientovi k zobrazení. Klient nemá možnost nahlédnout do jejich zdrojového souboru.

Pro vytváření programů, které spouští webový server, lze dnes využít mnoho technologií. Mezi dnes nejpoužívanější můžeme zařadit jednoduché skriptovací jazyky PHP a ASP (Active Server Pages), případně JSP (Java Server Pages) nebo CGI skripty. V následujícím textu bude ukázáno spojení WML s jazykem PHP.

PHP v současnosti zaznamenalo velký rozmach na internetu. Jeho nespornými klady jsou jednoduchost a snadná dostupnost. V případě použití nějakého jiného jazyka je začlenění WML velice podobné. V jednotlivých příkladech nastíním problematiku propojení PHP s WAPovým zařízením. Předpokládám částečnou znalost problematiky PHP.

4.1 Nastavení MIME typů

První věcí potřebnou pro běh PHP na počítači je nainstalování web serveru s podporou PHP. Nečastěji se používá Microsoft Personal Server, anebo Apache. Aby web server správně interpretoval WAPové soubory je potřeba nastavit jednotlivé MIME-types.

Tabulka č. 8 - MIME- types

Typ dokumentu	MIME type	Přípona Souboru
Jednoduchý WML dokument	text/vnd.wap.wml	.wml
Obrázek Wireless Bitmap	image/vnd.wap.wbmp	.wbmp
WMLScript	text/vnd.wap.wmlscript.wmls	.wmls
Kompilovaný WML dokument	application/vnd.wap.wml	.wmlc
Kompilovaný WMLScript	application/vnd.wap.wmlsrtiptc	.wmlsc

V případě, že nemáte přístup k modifikaci serveru, vloží se správný MIME-type do souboru s příponou .htaccess. Soubor se umístí v adresáři, který obsahuje WML dokumenty.
Příklad souboru .htaccess:

```
# MIME Types pro  WAP
# WML dokumenty.
AddType text/vnd.wap.wml .wml
# WML Wireless Bitmap.
AddType image/vnd.wap.wbmp .wbmp
# WMLScript.
AddType text/vnd.wap.wmlscript.wmls .wmls
# Kompilovaný WML dokument.
AddType application/vnd.wap.wml  .wmlc
# Kompilovaný WMLScript.
AddType application/vnd.wap.wmlsrtiptc .wmlsc
# Konec MIME Types
```

4.2 Generování stránky

Jak vygenerovat stránku s WML pomocí PHP ? Zjednodušeně řečeno, v praxi se to realizuje tak, že přikážeme kompilátoru PHP, aby příkazem pro výpis (typicky v PHP "echo") vypsal zdrojový text ve WML, který je pak stravitelný WAP-

browserem. WML kód je začleněn do PHP pomocí párových značek <? a ?>, které určují PHP kód.
Aby však WAP-browser věděl, že soubor končící příponou .php, nebo php3 obsahuje přeci jen kód WML, je nutné jej na to upozornit na prvním řádku následující hlavičkou:

```
<?
   header("Content-type: text/vnd.wap.wml");
   echo("<?xml version=\"1.0\"?>\n");
   echo("<!DOCTYPE wml PUBLIC \"-//WAPFORUM//DTD WML 1.1//EN\" \
  "http://www.wapforum.org/DTD/wml_1.1.xml\">\n\n");
?>
```

Příkaz “echo slouží pro výstup na obrazovku (respektive na displej), takže se vygeneruje klasická hlavička pro WML:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
  "http://www.wapforum.org/DTD/wml_1.1.xml">
```

Samozřejmě není nutné i samotnou hlavičku generovat dynamicky pomocí PHP. Lze ji zapsat i standardně ve WML. Takto zapsaná však nesmí být uzavřena mezi znaky <? a ?>, které signalizují následující PHP kód.

Ilustrační příklad stránky v PHP ukazující datum:

```
<?  Header("Content-Type: text/vnd.wap.wml");
    echo '<?xml version="1.0"?>';
?>
```

```
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
   "http://www.wapforum.org/DTD/wml_1.1.dtd">
<wml title="DATUM" >
 <? echo("<card id=\"jedinakarta\" title=\"anka\">\n");?>
   <p align="center">
  Dnes je: <?echo Date("d.m.Y")?><br/>
  Preji pekny den.
   </p>
  </card>
</wml>
```

V příkladu jsou použity dva prvky v PHP. Jedná se o již zmiňovaný příkaz echo a funkci date(), zobrazující aktuální čas. Na Mobilním telefonu by tato stránka vypadala následovně:

Obrázek č. 2 – Wap a PHP

4.3 Předávání WML proměnných v PHP

Hodnoty zadané uživatelem do vstupních polí na kartě, lze velice jednoduše předat skriptu běžícímu na serveru. Skript pak může na tyto hodnoty patřičně zareagovat.
Proměnné se předávají tak, že je přidáme za URL adresu skriptu za znak `**?**'. Kdybychom například chtěli předat obsah vstupního pole **o**, použijeme následující zápis:

Odkaz

Zápis $(promena) je v okamžiku aktivování odkazu nahrazen aktuálním obsahem proměnné. S takto předanou hodnotou pak můžeme pracovat v PHP skriptu. PHP nám pro každou předanou hodnotu vytvoří proměnnou odpovídající názvu.
V našem případě budeme mít v PHP skriptu k dispozici proměnnou $parametr. S ní pak můžeme naložit podle libosti.

Pokud potřebujeme skriptu předat více hodnot, stačí je v URL adrese odkazu oddělit znakem `**&**', který však musíme zapsat jako "&**;"**.
Následující příklad ilustruje využití. Z důvodů místa nejsou v příkladech hlavičky souborů.

```
<wml>
<card id="zadaniudaju" title="Zadani udaju" newcontext="true">
<p>
Hodnota1: <input format="*N" name="cislo1" title="Hodnota1:"/>
Hodnota2: <input format="*N" name="cislo2" title="Hodnota2:"/>
```

```
Operace: <select name="operace" value="secti" title="Operace">
<option value="secti">Scitani</option>
```

```
<option value="odecti">Odecitani</option>
</select>
<br/>
<anchor>Proved<go href="http://127.0.0.1/sdil3skript.php3?
cislo1=$(cislo1)&amp;amp;cislo2=$(cislo2)&amp;amp;operace=$(operace)"/>
</anchor>
</p>
</card>
</wml>
```

Skript na který směřuje odkaz z předcházejícího souboru:

```
<wml>
<card id="vysledek" title="Vysledek">
<p><?
$vysledek = 0;
Prvni hodnota: $cislo1 <br/>
Druha hodnota: $cislo2<br/>
if ($operace = = 'secti') {
$vysledek = $cislo1 + $cislo2;
$vybrano = 'Scitani';
} else  ($operace == 'odecti') {
$vysledek = $cislo1 - $cislo2;
$vybrano = 'Odecitani';
 }
echo("Operace: $vybrano");
<br/>
echo("Vysledek: $vysledek\n");
?></p>
</card>
</wml>
```

Na displeji mobilního telefonu bude tento program vypadat přibližně takto:

Obrázek č. 3 – kalkulačka v PHP

4.4 WAP a databáze

Nyní se podíváme na skutečně dynamické skriptování. Vzhledem k tomu, že WAP má být bohatým mobilním informačním zdrojem jako jsou databázové systémy. Lze tak velice snadno vytvořit velmi užitečné aplikace jako jsou různé podnikové informační systémy a přístup k internetovým službám. Skriptovací jazyk PHP nabízí knihovny funkcí pro práci s hezkou řádkou databází (Microsoft SQL, Oracle, SyBase, MySQL a řada dalších).

V dalším příkladu bude demonstrován přístup do databáze z WAPových stránek využívající databázového produktu MySQL. Samozřejmě že se může použít jiná databáze. Pro využití jiných databází je třeba prostudovat manuál k PHP pro nalezení funkcí spolupracujících s produkty jiných výrobců. Většinou mají velmi podobnou syntaxi.

Příklad bude vyhledávat záznamy z tabulky odpovídající zadanému příjmení a tyto záznamy následně zobrazí.Program obsahuje dvě stránky - zadání hodnot a zobrazení výsledků.

Úvodní strana se dotazuje na příjmení, podle kterého budou osoby vyhledávány:

```
<wml>
<card id="zadani" title="Zadani hledani" newcontext="true">
<p>
<?
echo("Zadej prijmeni: <input type=\"text\" name=\"klic\"/>\n");
echo("<anchor>Hledej zadani <go href=\
"sdil4skript.php3?klic=$(klic)\"/></anchor>\n");
?>
</p>
</card>
</wml>
```

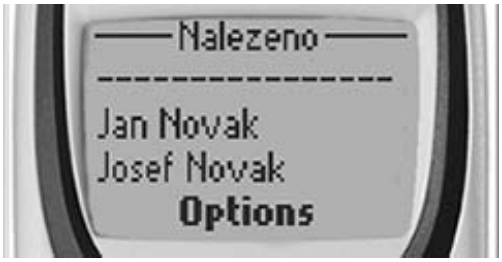
Odkaz pro vyhledání záznamů předává následujícímu skriptu proměnnou obsahující příjmení jako kritérium pro vyhledávání. A zde je text vyhledávacího skriptu:

```
<wml>
<card id="vysledek" title="Nalezeno">
<p>
<?
$konexe=mysql_connect('localhost','root','');
$navrat=mysql_db_query('jmena','SELECT
krestni,prijmeni FROM nazvy',$konexe);
echo("Vyhledavani dle: $klic");
echo("<br/>\n");
echo("Jmeno a prijmeni\n");
echo("<br/>\n");
echo("----------\n");
echo("<br/>\n");
while (list($krestni,$prijmeni) = mysql_fetch_row($navrat)) {
```

```
$prijmeni=trim($prijmeni);
if ($prijmeni == $klic)
{
echo("$krestni $prijmeni");
echo("<br/>\n");
}
}
echo("----------------\n");
echo("Konec vypisu\n");
?>
</p>
</card>
</wml>
```

Pro pochopení funkcí umožňujících spojení s MySQL databází opět odkazuji do manuálu PHP. Stěžejní část skriptu je pak v cyklu while, kde dochází k procházení záznamů z tabulky jmen a jejich kontrole, zda se hledané příjmení s nimi shoduje (viz opět manuál PHP).

Výsledkem pak bude výpis obsahující:



Obrázek č. 4 – Databáze na WAPu

4.5 Posílání dat z formulářů

Existují dvě základní metody posílání dat z formulářů, jedná se o GET a POST. Ty najdou své uplatnění také při tvorbě WAP stránek pomocí WML. Programátor zde však může narazit na řadu omezení a dalších skutečností, kterých si dosud nebyl vědom. Zjednodušeně řečeno, při použití metody GET jsou data předávána prostřednictvím URL, zatímco při využití POST nikoliv. Reálné mobilní telefony s WAPem mají totiž relativně výrazně omezenu maximální délku URL adresy. Toto omezení pak následně ovlivňuje maximální délku (součet délek) proměnných předávaných v URL adrese prostřednictvím metody GET (Nokia 7110 - 512 bytů). Zatímco u metody POST se metody nepředávají přímo URL adrese, je zde omezení pouze maximální velikostí decku ve zkompilované podobě (Nokia 7110 13000 bytů). Jednoduchý příklad na vytváření formulářů (buď GET, anebo POST) se skládá ze 2 částí kódu. První slouží k zobrazení formuláře, který předá proměnnou skriptu v PHP, jenž ji zobrazí.

```
<wml>
<card id="vstup" title="Vstup textu">
<p>Zadej text:<br/>
<input type="text" name="vstup"/>
    <anchor>Odeslat
    <go href="http://127.0.0.1/getipost.php3" method="post">
      <postfield name="text" value="$(vstup)"/>
    </go>
    </anchor>
</p>
</card>
</wml>
```

```
<? header("Content-type: text/vnd.wap.wml");
echo("<?xml version=\"1.0\"?>\n");
```

```
echo("<!DOCTYPE wml PUBLIC \"-//WAPFORUM//DTD WML 1.1//EN\"
 \"http://www.wapforum.org/DTD/wml_1.1.xml\">\n\n");
?>
<wml>
<card id="vystup" title="Zadano">
<p>
<? echo("$text");
?></p>
</card>
</wml>
```

V řadě aplikací jistě bude více využita metoda POST před GET, protože má menší omezení a nepředává uživatelské jméno a heslo přes URL. Samozřejmě záleží na programátorovi, pro kterou metodu se rozhodne.

4.6 Identifikace uživatele

Způsobů řešení tohoto problému existuje celá řada. Asi tou zcela nejjednodušší (pro „vývojáře“) je přinutit uživatele, aby se pro přístup na stránky zaregistroval a při jejich navštěvování se vždy přihlásil pomocí svého přiděleného uživatelského jména, popř. i hesla.
To však může uživatele do značné míry odradit od navštěvování takových stránek. Pomocí údajů z HTTP hlaviček předávaných spolu s uživatelovými požadavky na zobrazování WAP stránek jej však lze jednoznačně identifikovat zcela automaticky.

Příklad v PHP na následující straně ukazuje, jak takový identifikující údaj lze získat.

```
<?  header("Content-type: text/vnd.wap.wml");
```

```
 echo("<?xml version=\"1.0\"?>\n");
 echo("<!DOCTYPE wml PUBLIC \"-//WAPFORUM//DTD WML 1.1//EN\"
\"http://www.wapforum.org/DTD/wml_1.1.xml\">\n\n");
?>
<wml>
<card id="cislo" title="Telefonni cislo">
<p><?
echo("HTTP_X_UP_SUBNO: ".$HTTP_X_UP_SUBNO."<br/>\n");
?></p>
</card>
</wml>
```

Pomocí proměnných HTTP_X_UP_SUBNO by mohl server uživatele identifikovat. To však platí spíše obecně, ale rozhodně ne vždy. Pokud přistupujete z Paegas telefonu na Paegas gateway bude obsahovat proměnná řetězec (nejedná se o telefonní číslo), podle kterého můžete uživatele vždy identifikovat (např.2524761600_uwg.rdm.cz).

Přístup z EuroTel telefonu na EuroTel gateway zůstane proměnná prázdná. EuroTel gateway tento údaj totiž nevrací vůbec. Přístup na Paegas gateway by pro zákazníka patrně nebyl příliš cenově výhodný.

Z uvedeného je jasně vidět, že tento automatický způsob identifikace uživatele zatím ne vždy bude spolehlivě fungovat. Programátor, který se i přes tyto skutečnosti rozhodne něco podobného aplikovat do svých stránek, má v podstatě následující možnost. Pro uživatele Paegas může tento způsob použít a pro zákazníky EuroTelu bude muset použít způsob s využitím registrace a přihlašování uživatele.

4.7 Použití v praxi

Příklady měly za úkol ukázat tvůrci stránek základní techniky spojení WAPu s PHP. Nejedná se ovšem o manuál PHP, spíše jen o nastínění tohoto problému na daných ukázkách,
které dokazují, že propojení WAPu se serverovými aplikacemi má široké využití. Například si uživatelé mobilních telefonů s podporou WAP můžou zahrát šachy se svými přáteli (W@Pchess - http://nt.comtel.cz), nebo pomocí služby GCVoice (http://www.gcvoice.com) posílat hlasové zprávy jako e-mail. Samozřejmě těchto aplikací bude postupem času stále více a z mobilních zařízení se stanou nepostradatelní pomocníci v běžném životě.

5. Obrázky a animace

Stránky se dají obohatit o nějaký obrázek, který ovšem musí splňovat určitá kriteria vzhledem k daným omezením. Současná mobilní zařízení podporují pouze formát WBMP, který umožňuje ukládání pouze černobílých obrázků. V blízké budoucnosti by měl WAP podporovat i barevné obrázky jako je tomu na Japonských ostrovech (i-mode např. podporuje formát GIF).

Jak ale vytvořit, převést obrázek do formátu WBMP? Odpovědí se nabízí několik. Pokud má uživatel nainstalované vývojové prostředí Nokia WAP Toolkit 3, má o starost méně. V tomto prostředí si můžete nakreslit svůj vlastní obrázek, nebo zkonvertujete již existující s příponou jpg a gif. Na internetu lze naleznout mnoho programů pro konverzi, jako je WAP pictus a k většině grafických editorů se dá doinstalovat plug-in s podporou pro WBMP.

Musíme dbát na velikost obrázků. Většina mobilních telefonů je schopná zobrazit obrázky, jejichž šířka je okolo 95 pixelů a výška mezi 40 a 60 pixely.

Pro vložení do wapové stránky slouží element img. se dvěma atributy – src a alt, které jsou povinné. První určuje URL adresu obrázku a druhý textový popis. Je dobré uvést i atributy určující šířku a výšku obrázku (height a width), díky nimž může prohlížeč vynechat pro obrázek dostatečný prostor.

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
          "http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
 <card title="Stranka s obrazkem">
  <p align="center">
   <img alt="kacer" src="kachna.wbmp"
       height="50" width="37"/>
  </p>
 </card>
</wml>
```

Animovaného obrázku ve WAPu dosáhneme pomocí triku z časovačem. Jedná se o velice primitivní a nedokonalý způsob animace. Přesto může pomoci při tvorbě dynamických stránek.

Celý trik spočívá v postupném nahrávání obrázků jednoho za druhým. K tomu použijeme tagu TIMER a ONTIMER. Využití časovače při animaci popisuje následující zdrojový kód:

```
<?xml version="1.0"?>
 <!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN\"
 "http://www.wapforum.org/DTD/wml_1.1.xml">
 <!-Animace pomocí záměny souboru -->
 <wml>
  <card id=“obr1“ ontimer=“#obr2“>
    <timer value="9"/>
```

```
    <p  align=“center“>
      <a href="#c3">VSTUP</a>
      <img src=“srdce1.wbmp“ alt=“srdce1“/>
    </p>
  </card>
  <card id=“obr2“ ontimer=“#obr1“>
   <timer value="10"/>
    <p align=“center“>
      <a href="#c3">VSTUP</a>
      <img src= “srdce2.wbmp“ alt=“srdce2“/>
    </p>
  </card>
  <card id=“c3“ name=“Konec“>
       <p align=“center“>
        Konec animace.
       </p>
  </card>
 </wml>
```

Uživatel může narazit na problém s některými telefony (např. Nokia 7110), protože načítání stránky prokládají nápisem "Připojuji službu...". Tento nápis, pokud je zobrazen při načítání, pak přirozeně efekt celé animace kazí.

6. Čeština na WAPu

Pokud chcete tvořit české WAPové stránky s diakritikou máte v podstatě dvě možnosti. Prvním méně pohodlným způsobem je psaní českých znaků pomocí znakových entit. Použijeme zápis &#xkód;.
Kód udává daný znak, který bude vložen do stránky. Kódy jednotlivých znaků lze zjistit například na adrese http://www.unicode.org/charts/PDF/U0080.pdf.

V případě psaní delšího textu doporučuji využít programy na konverzi těchto znaků, Jeden z nejznámějších on-line překladačů češtiny je od společnosti PC Net CZ a nachází se na internetové adrese „http://www.pcnet.cz/wap2cz.phtml."

Tabulka č. 9 – Entity českých znaků

Entity českých znaků			
ě	&#x011B;	Ě	&#x011A;
š	&#x0161;	Š	&#x0160;
č	&#x010D;	Č	&#x010C;
ř	&#x0159;	Ř	&#x0158;
ž	&#x017E;	Ž	&#x017D;
ý	&#xFD;	Ý	&#xDD;
á	&#xE1;	Á	&#xC1;
í	&#xED;	Í	&#xCD;
é	&#xE9;	É	&#xC9;
ú	&#xFA;	Ú	&#xDA;
ů	&#x016F;	Ů	&#x016E;
ó	&#xF3;	Ó	&#xD3;
ť	&#x0165;	Ť	&#x0164;
ň	&#x0148;	Ň	&#x0147;
ď	&#x010F;	Ď	&#x010E;

Druhou možností je použít jiné kódování než je UTF-8. Použijeme-li kódování ISO-8859-2, které se používá především na unixových systémech máme víceméně vyhráno. Oba čeští operátoři si s tímto kódováním poradí.

Příklad kódování pomocí ISO-8859-2:

```
<?xml version="1.0" encoding="iso-8859-2"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
<card title="CS v iso-8859-2">
<p align="center">Červeňoučké jablíčko.</p>
</card>
</wml>
```

Nabízí se ještě možnost užít kódování windows-1250, které je ovšem bez záruk, nicméně také funguje.

7. Nastoupí XHTML ?

Nově připravovaná verze WAP 2.0 sebou přinese mnoho změn. Základní změnou bude bezesporu podpora jazyka XHTML, který by měl plně nahradit jazyk WML. XHTML má stejně jako WML své kořeny v obecném jazyce XML. Mnoho uživatelů napadne otázka proč se nástupcem nestalo klasické HTML, které již mají programátoři zažité v krvi?

Odpověď spočívá ve snadnějším rozšiřování jazyka o další prvky. HTML vychází z poněkud odlišného a také podstatně staršího standardu SGML, u kterého rozšiřování není tak efektivní jako u jazyků založených na XML. Stránku v XHTML si uživatel bude moci prohlédnout na mobilním zařízení i na klasickém internetu.
XHTML má ve srovnání se současným WML samozřejmě podstatně širší možnosti, kterými se vyrovná klasickým internetovým prohlížečům - multimediální aplikace, přehrávání hudebních souborů, video, streaming atd. Jazyk WML a WMLScript bude zpětně podporovat i WAP 2.0 .

Bohužel zatím není známo, kdy se na novější verzi přejde. Zatím na trhu neexistuje ani jeden mobilní telefon s podporou WAP 2.0. Podle mého názoru současný WAP zatím plně dostačuje požadavkům většiny zákazníků a mobilní operátoři příliš do nové technologie nepospíchají. Takže budeme muset na nové změny ještě nějaký čas počkat.

8. Závěr

Co dodat závěrem? Mnoho tvůrců narazí na základní problém v nekompatibilitě některých prohlížečů. Co jedny telefony podporují, druzí mohou ignorovat. Například u Nokia telefonů je zbytečné se snažit formátovat písmo, nebo tabulku. Dalším podstatným problémem je nedostatečná podpora některých funkcí WMLScriptu. Takže můžete napsat krásnou aplikaci ve svém vývojovém prostředí, která v reálném provozu bude k ničemu. Velmi dobrou alternativou je používání skriptovacích jazyků jako je PHP či ASP, ale pro jednoduché věci je tento způsob dosti neefektivní.

Nicméně spousta aplikací ve WAPu již dnes docela dobře funguje. Příkladem mohou být různá vyhledávání vlakových a autobusových spojů, přístupy k podnikovým informacím, či posílání emailu. WAP si své místo na trhu již více či méně nalezl a jeho základní rysy kopírují omezení a vlastnosti veškerých technologií, kterých pro svou činnost využívá. Předpokládaná verze WAP 2.0 je svým komfortem s dosavadní verzí zcela nesrovnatelná. Svými schopnostmi se již znatelně přibližuje možnostem prohlížečů na osobních počítačích. WAP je tedy v neustálém vývoji. Na přehrávání videosekvencí na displeji mobilního telefonu si zřejmě budeme muset ještě hodně dlouho počkat. Ale jsem si jist, že ta doba nastane.

Použité zdroje

České zdroje:

- http://www.interval.cz
- http://www.kosek.cz
- http://www.mobil.cz
- http://www.wmarks.cz
- http://www.wapdev.sk
- http://www.wapserver.cz
- http://www.wapway.cz

Cizojazyčné zdroje:

- http://www.ericsson.com
- http://www.gelon.net
- http://www.nokia.com
- http://www.w3schools.com
- http://www.wapforum.com
- http://www.wirelessdevnet.com
- http://www.wmlscript.com

PŘÍLOHY

Příloha č. 1

Nastavení WAPu podle poskytovatele služeb

Názvy jednotlivých položek se liší telefon od telefonu, i když je vyroben stejným výrobcem. Každý mobil také nevyžaduje všechna nastavení uvedená v tabulce níže.

Nastavení WAPu	EuroTel	*T-Mobile*	Oscar
Domovská stránka	http://wap/main.wml	http://wap	http://wap
Druh spojení	Trvalé spojení	Trvalé spojení	Trvalé spojení
Zabezpečení spojení	Vypnout	Vypnout	Vypnout
Nosič	Data	Data	Data
Vytáčené číslo	+420 602 900 927	+420 603 124 927	+420 608 989 896
Adresa IP	160.218.190.218	010.000.000.010	010.011.010.011
Typ autentizace	Normální	Normální	Normální
Typ datového volání	ISDN	ISDN	ISDN
Rychlost datového volání	14,4 kbps	9,6 kbps	9,6 kbps
Jméno uživatele	EuroTel	wap	wap
Heslo	wap	wap	wap

Příloha č.2.

Příklad odeslání emailu s kontrolou dat

Tento příklad má tří části:

1. **mail.wml**, který obsahuje tři karty(pro zadání dat, odeslání a oznámení chyby v adrese).
2. **kontrola.wmls** (kontroluje zadanou emailovou adresu)
3. **odeslat.php** (odeslání pošty).

mail. wml

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC"-//WAPFORUM//DTD WML 1.1//EN"
   "http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>
  <card id="vstup" title="posta">
    <p>Od: <input type="text" name="od" format="*M"/></p>
    <p>Komu: <input type="text" name="adresa" format="*M"/></p>
    <p>Predmet: <input type="text" name="predmet" format="*M"/></p>

     <do type="accept" label="Odeslat">
          <go href="kontrola.wmls#kontrola('$(adresa)')"/>
     </do>
  </card>

  <card id="error" title="Chyba ve vstupu">
          <p>$(email)</p>
          <p>   <a href = "#vstup">zadat znovu</a></p>
   </card>
  <card id="odeslat" title="posta">
```

```
    <p>
      <anchor>Odeslat email
       <go method="post" href="odeslat.php?action=send">
          <postfield name="od" value="$(od)"/>
          <postfield name="komu" value="$(adresa)"/>
          <postfield name="predmet" value="$(predmet)"/>
           <postfield name="zprava" value="$(zprava)"/>
       </go>
      </anchor>
    </p>
  </card>
</wml>
```

kontrola.wmls

```
extern function kontrola(adresa) {
     var chyba=false;
if (String.length(adresa)<6) {
          WMLBrowser.setVar("email","chybna adresa");
          chyba=true;
}
if(String.find(adresa,"@") = = (-1)) {
 WMLBrowser.setVar("email","email neobsahuje @");
  chyba=true;
}
if (chyba) {  //návrat na příslušnou kartu
  WMLBrowser.go("mail.wml#error");
} else {
  WMLBrowser.go("mail.wml#odeslat");

}
};
```

odeslat.php

```
<?
  header("Content-type: text/vnd.wap.wml");       // hlavička pro kód WML
  echo("<?xml version=\"1.0\"?>\n");
  echo("<!DOCTYPE wml PUBLIC \"-//WAPFORUM//DTD WML 1.1//EN\"
\"http://www.wapforum.org/DTD/wml_1.1.xml\">\n");

  $server = "romeo.pf.jcu.cz";                    // jméno poštovního serveru
  $od = $od." (Uzivatel na: ".$server.")";



  echo("<wml>\n");                                 // Odeslání
  if(mail($adresa, $od,$predmet,$zprava) {         // využití PHP funkcí
    echo("<card id=\"odeslano\" title=\"odeslano\">\n");    // posta byla odeslána
    echo("<p>Posta byla odeslana</p>\n");
    echo("</card>\n");
  }
  else {
    echo("<card id=\"zavada\" title=\"neodeslano\">\n");   // pošta neodeslána
    echo("<p>Nemohu odeslat postu</p>\n");
    echo("</card>\n");
  }
  echo("</wml>\n");
?>
```

Příloha č. 3

Algoritmus bubble-sort ve WMLScriptu

Příklad seřazení zadaných čísel pomocí algoritmu bubble-sort demonstruje možnost využití práce s řetězci namísto datového typu pole.

bubble.wmls

```
extern function bubble(retezec){
  var vysledek = false;
  var i;
  var temp;

  while (!vysledek){
    vysledek = true;
    for (i = 0 ; i < 5; i++) {
      var a = String.elementAt(retezec,i,",");
      var b = String.elementAt(retezec,i+1,",");
      if (a > b) {
        temp = String.elementAt(retezec,i+1,",");
        retezec = String.replaceAt(retezec,a,i+1,",");
        retezec = String.replaceAt(retezec,temp,i,",");
        vysledek  = false;
      }
    }
  }
  WMLBrowser.setVar("retezec",retezec);
  WMLBrowser.go("bubble.wml#vysl");
}
```

bubble.wml

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>
<card id = "bubble" title = "Bubble Sort">
  <onevent type = "onenterforward">
        <refresh>
        <setvar name = "retezec" value = ""/>
        </refresh>
  </onevent>
<p>
  Zadejte 5 cilel oddelenych carkou v rozmezi 0 - 99:<br/>
  <input name = "retezec"/><br/>
  <a href = "bubble.wmls#bubble('$(retezec)')">Seradit</a>
</p>
</card>

<card id = "vysl" title = "Vysledek">
<p>
  Vysledek je:<br/>
</card>
</wml>
```

Příloha č. 4

Vytáčení telefonního čísla přes WAP

Pokud na svých WAPových stránkách uvádíte telefonní číslo, můžete pomocí funkce URI zajistit, aby návštěvníkovi stačilo jediné kliknutí pro zavolání nebo uložení čísla do seznamu.

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>
  <card id = "kontakt" title = "Kontaktujte nas">
    <p>
      <b>Kontakt</b><br/>
       servisni čislo: <br/>
       <a href="wtai://wp/mc;+420628875662">0628875662</a><br/>
       ulozit cislo do seznamu: <br/>
      <a href=" wtai://wp/ap;+420628875662, "Kuba Janek">0628875662</a><br/>
   </p>
 </card>
</wml>
```

Příloha č. 5

Přehled mobilních telefonů s podporou WAPu na našem trhu

Alcatel One Touch 301
Alcatel One Touch 302
Alcatel One Touch 303
Alcatel One Touch 311
Alcatel One Touch 501
Alcatel One Touch 511
Alcatel One Touch 701
Benefon Q
Ericsson A2628s
Ericsson A3618s
Ericsson R320s
Ericsson R380e
Ericsson R380s
Ericsson R520m
Ericsson T20e
Ericsson T20s
Ericsson T29s
Ericsson T39m
Ericsson T65
Ericsson T68
Motorola Acc. 008
Motorola T2288
Motorola T2288R
Motorola Talk. 191
Motorola Talk. 192
Motorola Talk. 205
Motorola Timeport 250
Motorola Timeport 260
Motorola Timeport 280
Motorola Tim. P7389
Motorola V.100
Motorola V.50
Motorola V.60
Motorola V.66
Nokia 3330
Nokia 5210
Nokia 5510
Nokia 6210
Nokia 6250
Nokia 6310
Nokia 6510
Nokia 7110
Nokia 7650
Nokia 8250
Nokia 8310
Nokia 9110i
Nokia 9210
Panasonic GD 35
Panasonic GD 75
Panasonic GD 93
Panasonic GD 95
Philips Azalis 238
Philips Azalis 288
Philips Fisio 311
Philips Savvy Vogue
Sagem WA3050
Samsung SGH-N100
Samsung SGH-N300
Samsung SGH-Q100
Siemens C35i
Siemens C45
Siemens M35i
Siemens ME45
Siemens S35i
Siemens S40
Siemens S45
Siemens SL45(i)
Sony CMD-J5
Sony CMD-J6
Sony CMD-J7
Sony CMD-J70
Sony CMD-MZ5
Sony CMD-Z5
Sony CMD-Z7
Trium Aria-@
Trium Eclipse

Příloha č. 6

Slovníček anglických zkratek

DTD - Document Type Definition
DTMF - Dual Tone Multi-Frequency
GPRS - General Packet Radio Service
GSM - Global System for Mobile Communication
HTML - HyperText Markup Language
HTTP - Hypertext Transfer Protocol
IP - Internet Protocol
ISO - International Organization for Standardization
PAP - Push Access Protocol
PCI - Protocol Control Information
PCS - Personal Communication Services
PDA - Personal Digital Assistant
PDC - Personal Digital Cellular
PI - Push Initiator
PIN - Personal Identification Number
PPG - Push Proxy Gateway
SGML - Standard Generalized Markup Language
SIM - Subscriber Identity Module
SMS - Short Message Service
SMTP - Simple Mail Transfer Protocol
SSL - Secure Socket Layer
TCP/IP - Transmission Control Protocol/Internet Protocol
UI - User Interface
URI - Uniform Resource Identifier
URL - Uniform Resource Locator
UTF - UCS Transformation Format
W3C - World Wide Web Consortium
WAE - Wireless Application Environment
WAP - Wireless Application Protocol
WBMP - Wireless BitMaP
WBXML - WAP Binary XML
WML - Wireless Markup Language
WMLScript - Wireless Markup Language Script
WSP - Wireless Session Protocol
WTA - Wireless Telephony Application
WTAI - Wireless Telephony Application Interface
WTLS - Wireless Transport Layer Security
WTP - Wireless Transaction Protocol
WWW - World Wide Web
XHTML - Extensible HyperText Markup Language
XML - Extensible Markup Language