Czym jest XMERL ?

Xmerl jest zbiorem modułów, które mogą posłuzyć do łatwego parsowania plików xml zgodnych ze standardem 1.0.

Xmerl składa się z kilku tylko modułów

xmerl – funkcje do eksportowania danych xml do innych formatów,
na przykład html, txt...

Xmerl składa się z kilku tylko modułów

xmerl_scan – ten moduł zawiera funkcje służące parsowaniu plików xml

Xmerl składa się z kilku tylko modułów

xmerl_xpath – ten moduł wspiera całą specyfikację Xpath w wersji 1.0

Xmerl składa się z kilku tylko modułów

xmerl_eventp – Parser SAX
(Simple API for XML)

Xmerl składa się z kilku tylko modułów

xmerl_xs – *Moduł do transformacji pliku xml do danego formatu, coś na wzór XSLT (Extensible Stylesheet Language Transformations)*

EPL – czyli Erlang Public License

- Oparta na Mozilla Public License
- Fragmenty prawne związane z prawem Szwedzkim
- Publikowana wraz z „angielskim tłumaczeniem tekstu prawnego”
- http://www.erlang.org/EPLICENSE

EPL – co dla programisty?

- Kod źródłowy rozpowszechniany przez Ericsson w celu promocji Erlanga
- Otwarty na modyfikację i rozszerzenia
- Powiązany kod musi być udostępniany Open Source wraz z plikiem licencyjnym
- „We are not giving you any guarantees. If Erlang doesn't work or it causes you damage in any way, tough luck!”

Żywot Xmerla

- Pierwsze wersje pod koniec 2001r.
- W latach 2002/2003 pozornie niewiele zmian: wersje 0.17 – 0.19 …
- … jednak v. 0.20 z 2004 roku szybko stała się release 1.0
- Release notes: http://erlang.org/doc/apps/xmerl/notes.html

A po co ten Xmerl?

- Pierwotnym celem było rozwijanie wsparcia Erlanga dla standardu WAP.
- Ostatecznie wspartych zostało tylko kilka elementów standardu… w tym obsługa XML.
- Przez społeczność Open Source Xmerl został potraktowany jako uniwersalne narzędzie do wymiany danych – poprzez pliki XML.

Co dobrego?

- Relatywnie prosta obsługa w kodzie
- Uniwersalny interfejs wymiany danych poprzez pliki XML
- Niektóre standardy wymagają obsługi plików XML
- Może być wykorzystany do konwersji XML na inne typy
- W związku z EPL – pełny open source: http://sowap.sourceforge.net/

Co może nie być dobre?

- W Xmerlu działa filozofia „Let it crash”, o czym należy pamiętać przy wykorzystaniu go we własnych modułach.
- Xmerl generalnie obsługuje kodowanie Unicode, jednak nie radzi sobie ze znakami, które nie są mapowane na ASCII
- Napotkanie takiego znaku podczas parsowania nie jest dobrze zdefiniowane (znak może zostać pominięty… albo spowodować crash).

Widziane na forach…

- Najczęściej zgłaszanym problemem jest nieścisłość dokumentacji
- Przeszkadzają także niejasno określone zachowania przy parsowanu niektórych kodowań
- Od 2009 roku użytkownicy czekają na zdefiniowanie zachowania przy parsowaniu Unicode (propozycja: pozostawić znak bez zmian, nie mapować na ASCII).

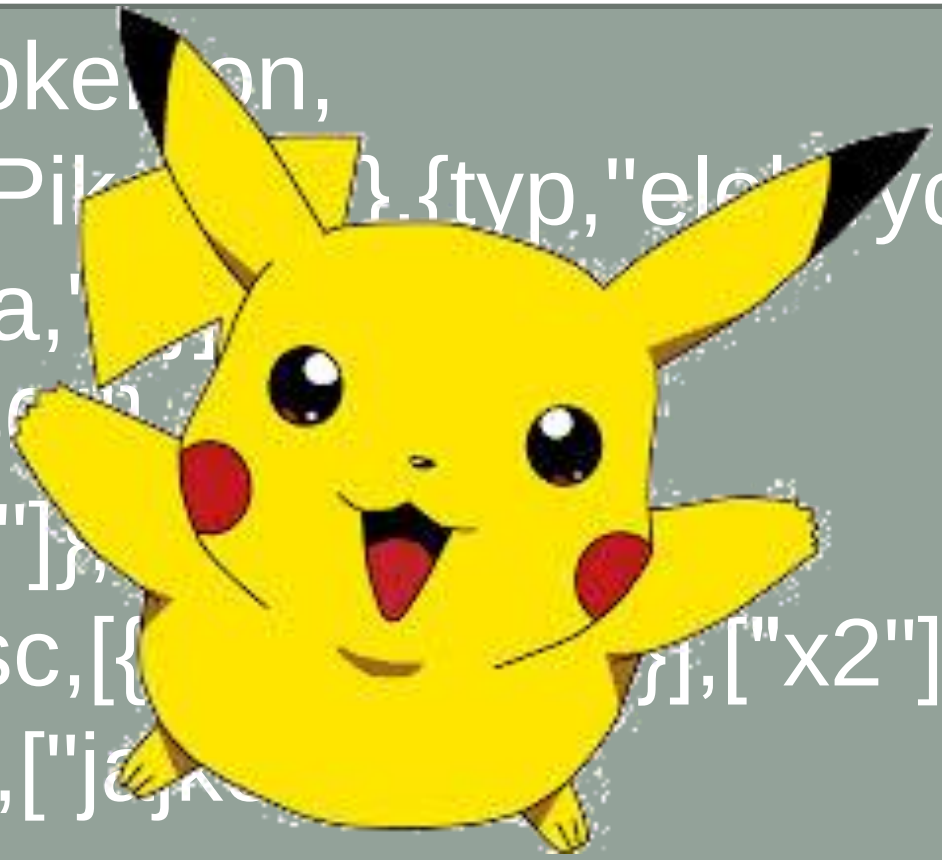
Wewnętrzna reprezentacja XMLa

{TAG,
LISTA ATRYBUTÓW,
LISTA Z ZAWARTOŚCIĄ}

Można ominąć listę atrybutów
i listę z zawartością

Wewnętrzna reprezentacja XMLa

```
Pika = {poke          on,
[{nazwa,"Pik          },{typ,"ele          yczny"},
{generacja,"          ]
[{zycie,["6(          ]
{atak,["10"]},
{podatnosc,[{          }],["x2"]},
{ewolucja,["ja          k
```



ERLANG

Wewnętrzna reprezentacja XMLa

Przykładowe rekordy mogące ułatwić pracę

- #xmlText
- #xmlElement
- #xmlPI
- #xmlComment
- *#xmlDecl*

Wewnętrzna reprezentacja XMLa

Dołączamy nagłówek, by mieć dostęp do rekordów:

-include_lib("xmerl/include/xmerl.hrl").

rr(xmerl) - w shellu

Otwieranie XMLa

```
{File,Misc} =
        xmerl_scan:file("pokemony.xml",
        [{validation,true}]).
```

Zapisywanie

```
XMLFile = xmerl:export_simple(XmlVariable, xmerl_xml).
XMLUNI = unicode:characters_to_binary(lists:flatten(XMLFILE)).
{ok,FH} = file:open("plik.xml",[write]).
io:format(FH,"~s~n",[XMLUNI])
```

Xpath - przykład

```
%wyszukiwanie
Result = xmerl_xpath:string("/pokemony//*[@typ]", File).
```

```
%podmienianie wartości atrybutu
[A] = xmerl_xpath:string("/pokemony/pokemon//@typ[.='ogniowy']",
File).
B = A#xmlAttribute{value = 'wodny'}.
[C] = xmerl_xpath:string("/pokemony//pokemon[@typ='ogniowy']",
File).
#xmlElement{attributes = [A1,A2,_]} = C.
D = C#xmlElement{attributes = [A1,A2,A]}.
```

Jedna z możliwości dodania elementu

```
#xmlElement{content = C} = File.
NewC = C++[NewNode].
NewFile = File#xmlElement{content = NewC}.
```

That’s not it! …yet

Maksymilian Gasztych i Jakub Pilch