

# Юникод in Rails

the how's and why's

# Who

- Julian “Julik” Tarkhanov
- from Moscow
- doing my bachelor in Utrecht
- and web sites
- 4 languages, 3 of them in daily use

# Why?

- I am Russian and I want to use apps that I am making
- My clients for the most part are Russian
- ...and Dutch
- ...and sometimes even Israelian

“...one thing I’ve learned from six years working in commercial IT in Germany is that the first thing you do to any prospective vendor’s software is throw a few umlauts in, then laugh and send them away when it immediately collapses in a heap.”

<http://www.alanlittle.org/weblog/OnLanguages.html>

# What is Unicode?

- A way to store, transform and represent all modern writing systems digitally
- That's how you store and handle the string “уехал в København” in one place
- In Unicode, a character != a byte
- And a codepoint is not a character

# Nuts and bolts

- Abstract codepoints ( A Ш ¥ ¢ )
- Modifier codepoints ( ° ^ )
- Direction control codepoints (RTL-LTR override)
- 26 kinds of whitespace
- “smart quotes”

# Non trivial

"efficient"[0..1]  $\#=>$  "ef" ?

Hint: `ffi` is one codepoint and is matched as a character

# Sad truth

- The only 2 languages/environments getting it right are C# and Java
- They both enforce UTF and it's marvelous
- ...yes, Python, PHP and Perl suck at it too
- ...and you can screw it up in Java remarkably easily shall you be clueless

# Unicode in Ruby

- Non-existent, \$KCODE is just for parsing sources and regexen, and incomplete ones at that. Oniguruma is still not default.
- Matz promises Unicode among other M17N features in 2007 and he's not a fan of Unicode
- ...my last Rails application is due yesteryear

# Ruby eats babies

"Café".length # => 5 - oops...

"Café".reverse # => faC - oh my!

"Café"[0..3] #=> Caf\303 - ouch!

"efficient".split(//)

#=> ["e", "ffi", "c", "i", "e",  
"n", "t"] - nooo....

# The best part

## Try Ruby!

```
Interactive ruby ready.
```

```
>> "ТЕКСТ".length
```

```
=> 30
```

```
>>
```

```
>>
```

```
=> 30
```

# Making RoR do Unicode and II8N



# Bondage

- Globalize guys are the only ones who care
- even no access to Locale
- ...and this Unicode thing

# Remaining bits of joy

- Unicode and mbstring extensions by Yoshida Masato - unfortunately obsolete
- 7 years old!
- does casefolding but normalization is now invalid

# Overrides: unicode\_hacks

- based on the Unicode extension
- overrides ALL of the String#methods
- and makes them work
- and breaks other stuff
- CGI.escape, ERB compiler, Webrick...

# Why stuff breaks?

- In PHP you don't notice because all the low-level stuff happens in C
- Content-length?

# Solution II: ICU4R

- Marvelous extension by Nikolai Lugovoi based on ICU by IBM
- Actually a golden standard
- somewhat incompatible with Ruby strings
- Uses UTF-16 instead of UTF-8 used in Ruby

# Subclassing: The Python way

- `...encode(decode(recode("hopeless crap")))`
- your ordinal is still not in range!

# Solution III: Accessor proxy

```
"Café".chars.length # => 4
```

```
"Café België".chars.reverse  
#=> "ëigleB éfaC"
```

```
"efficient".chars.split(//)  
#=> ["e", "f", "f", "i", "c",  
"i", "e", "n", "t"]
```

# Much nicer

Chainable, comparable, sortable

```
"Café".chars.reverse.length # => 4
```

```
"Café".chars == "Café" #=> true
```

```
"\100\110\255".chars.length #=> 3
```

# Composition and normalization

- A process of removing alternate representations of equivalent sequences from textual data, to convert the data into a form that can be binary-compared for equivalence

<http://diveintomark.org/archives/2004/07/06/nfc>

# Seemingly non-trivial issues

- Normalization - is the ligature ffi really one character? **No thanks... not in my Rails app**
- Composition - Is Ø really two characters? **Hell no... not in my app either**

Thus I vote for **NFKC** and I `pre_filter` my  
params

[http://julik.textdriven.com/  
svn/tools/rails\\_plugins/  
unicode\\_hacks](http://julik.textdriven.com/svn/tools/rails_plugins/unicode_hacks)

normalize\_params

String#chars

more to come

Please try it and tell me where it breaks - [me@julik.nl](mailto:me@julik.nl)

# Watch out

- the “i” modifier in Regexp is not Unicode-aware, even with Oniguruma in 1.9
- XML Builder escapes all Unicode characters
- YAML converts Unicode strings to base64

# Please DO care!

- There is no such thing as plain text
- Do patches and extensions
- Write detailed, extremist emails and whine relentlessly
- Test that stuff
- Have at least one non-English/Dutch/European developer/tester on the team if you can
- Help the `String#chars` accessor get into Rails proper - we are making it into a patch soon

# Resources

<http://po-ru.com> - Paul Battley speaks Japanese and does RoR full-time, and is fully aware of where Unicode fails (from Unicode perspective). Unfortunately his presentation was on my birthday in Munich so I couldn't attend

<http://blogamundo.com> - A blog focusing on the issues more broadly, with your daily Loonicode strip

# Thank you

NB: All the insulting statements have been made intentionally, as the insulted parties have deserved

## Questions?