

Polyglot Programming

Barney Boisvert

www.barneyb.com

bboisvert@gmail.com

@barneyb

Who Am I?

- Sr. Application Engineer at Mentor Graphics
- Framework Author
- Code tinkerer
- Blogger
- Father
- Motorcyclist
- Tall
- etc.

What am I talking about?

- Using Languages
- Combining Languages
- Creating Languages
- Helping you be lazy like me

What is Polyglot Programming?

What is Polyglot Programming?

```
#define a /*
#<?php
echo "\010Hello, world!\n"; // 2> /dev/null > /dev/null \ ;
// 2> /dev/null; x=a;
$x=5; // 2> /dev/null \ ;
if (($x))
// 2> /dev/null; then
return 0;
// 2> /dev/null; fi
#define e ?>
#define b */
#include <stdio.h>
#define main() int main(void)
#define printf printf(
#define true )
#define function
function main()
{
printf "Hello, world!\n"true/* 2> /dev/null | grep -v true*/;
return 0;
}
#define c /*
main
#*/
```

From [http://en.wikipedia.org/wiki/Polyglot_\(computing\)](http://en.wikipedia.org/wiki/Polyglot_(computing))

I'm kidding.

The Real Deal

- Low-level languages:
 - hard for people
 - easy for computers
- High-level languages:
 - easy for people
 - hard for computers
- Domain Specific Languages (DSLs)
 - easy for people
 - purpose (domain) specific

Manage the gradients!

Manage the What?

- Pick the right language
 - readability
 - intent, not just parsing
 - encapsulation
 - maintainability
 - reuse
- Design your APIs
 - method/variable names
 - URL structures
 - Database structures

A Simple Example

- Render a sponsor list, grouped by level.
- Items have a link to the sponsor's site
- Using ORM for persistence

Mercury

- [Sleazy Metals](#)
- [Metal Shift](#)

Platinum

- [B2 Metalurgy](#)
- [B2 Metals](#)

Titanium

- [A1 Metallics](#)
- [A1 Hard & Shiny](#)

ALL CFML

```
<cfset sponsors = ormExecuteQuery("
  from Sponsor
  order by level.name, name
") />

<cfset lastLevel = "" />
<cfloop array="#sponsors#" index="sponsor">
  <cfif sponsor.getLevel().getName() NEQ lastLevel>
    <h2>#level#</h2>
    <ul>
  </cfif>
  <li><a href="#sponsor.getWebsite()#">#sponsor.getName()#</a></li>
  <cfif sponsor.getLevel().getName() NEQ lastLevel>
    </ul>
    <cfset lastLevel = sponsor.getLevel().getName() />
  </cfif>
</cfloop>
```

All CFML Too

```
<cfset sponsors = ormExecuteQuery("from Sponsor order by name") />
<cfset sponsorsByLevel = {} />
<cfloop array="#sponsors#" index="sponsor">
  <cfset level = sponsor.getLevel().getName() />
  <cfif NOT structKeyExists(sponsorsByLevel, level)>
    <cfset sponsorsByLevel[level] = [] />
  </cfif>
  <cfset arrayAppend(sponsorsByLevel[level], sponsor) />
</cfloop>

<cfloop collection="#sponsorsByLevel#" item="level">
  <h2>#level#</h2>
  <ul>
    <cfloop array="#sponsors[level]#" index="sponsor">
      <li><a href="#sponsor.getWebsite()#">#sponsor.getName()#</a></li>
    </cfloop>
  </ul>
</cfloop>
```

A Bit of Groovy

```
<cfset sponsors = ormExecuteQuery("from Sponsor order by name") />
<g:script>
    variables.sponsorsByLevel = variables.sponsors.groupBy {
        it.getProperty('level').getProperty('name')
    }
</g:script>

<cfloop collection="#sponsorsByLevel#" item="level">
    <h2>#level#</h2>
    <ul>
        <cfloop array="#sponsors[level]#" index="sponsor">
            <li><a href="#sponsor.getWebsite()#">#sponsor.getName()#</a></li>
        </cfloop>
    </ul>
</cfloop>
```

More Groovy Magic

```
<cfset sponsors = ormExecuteQuery("from Sponsor order by name") />
<g:script orm="true">
    variables.sponsorsByLevel = variables.sponsors.groupBy {
        it['level']['name']
    }
</g:script>

<cfloop collection="#sponsorsByLevel#" item="level">
    <h2>#level#</h2>
    <ul>
        <cfloop array="#sponsors[level]#" index="sponsor">
            <li><a href="#sponsor.getWebsite()#">#sponsor.getName()#</a></li>
        </cfloop>
    </ul>
</cfloop>
```

The `orm="true"` Magic

```
@Category(coldfusion.runtime.TemplateProxy)
class CfOrmCategory {
    Object getAt(String prop) {
        this.getProperty(prop)
    }
    Object putAt(String prop, Object value) {
        this.setProperty(prop, value)
    }
}
use (CfOrmCategory) {
    // script goes here
}
```

Language Selection

- the "main" language
- the "helper" language(s)
 - features
 - performance
 - readability/maintainability
- the "library" language(s)
 - third party
 - legacy
- the domain specific language(s)

DSLs...

- Are domain specific (duh)
- Vary widely in complexity
 - a few related methods on one type
 - a compiler, IDE, runtime, etc.

DSL Examples

- SQL

DSL Examples

- SQL
- Google Search

DSL Examples

- SQL
- Google Search
- xUnit

DSL Examples

- SQL
- Google Search
- xUnit
 - Hamcrest

DSL Examples

- SQL
- Google Search
- xUnit
 - Hamcrest
- Ant

Creating DSLs

1. Domain

- nouns
- verbs
- semantics

2. Users

- skill level
- nomenclature
- environment

3. Implementors

- that's YOU!

Wrapping Up

- Don't just use some language, pick your language
- Developers *constantly* create new languages
- "Lazy" is a badge of honor. And you have to earn it.

Soapbox

- If you're not using source control, don't write another line of code until you start.
- Don't build another application without a public front controller framework.
- Your tools are WAY more capable than you (or I) know. Invest time to learn about them.

Barney Boisvert

www.barneyb.com

bboisvert@gmail.com

@barneyb