Writing Domain Specific Languages in Boo

Oren Eini
We!
ayende@ayende.com
http://www.ayende.com/Blog/

Who am I?

- Developer
- Blogger
- Work: We! Consulting Group
- Also:
  - Your entertainment for the next hour or so

Who are **you**?

- Do you know what a DSL is?
- Wrote a DSL?
- Used a DSL?

Speaking in tongues

How was work today, honey?

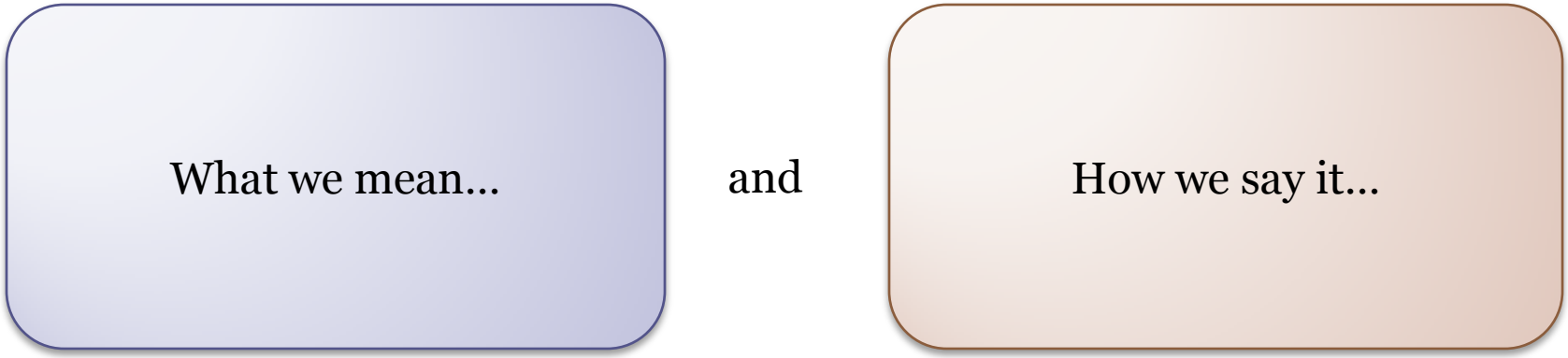
I have implemented a caching proxy and configured it with runtime XML… blah, blah, blah

Now he is starting to talk in gibberish again. Oh, if only we could *talk*…

```
push    edi                         ; function save registers
push    esi
mov     edi, dword [esp + 0Ch]      ; get the pointer to string
sub     eax, eax                    ; look for zeros
sub     ecx, ecx
dec     ecx                         ; set ecx to -1
repnz   scasb                       ; search for 0 in string
neg     ecx
sub     ecx, 2                      ; get the string length w/o zero
mov     esi, dword [esp + 0Ch]      ; get pointer once again
.putchar_loop:
push    ecx                         ; keep the counter
lodsb                               ; get the char
push    eax
call    _putchar                    ; print char to stdout
add     esp, 4                      ; correct stack
pop     ecx                         ; get back the counter
dec     ecx
jnz     .putchar_loop               ; if not last char then get next
pop     esi                         ; restore registers
pop     edi
retn    4
```

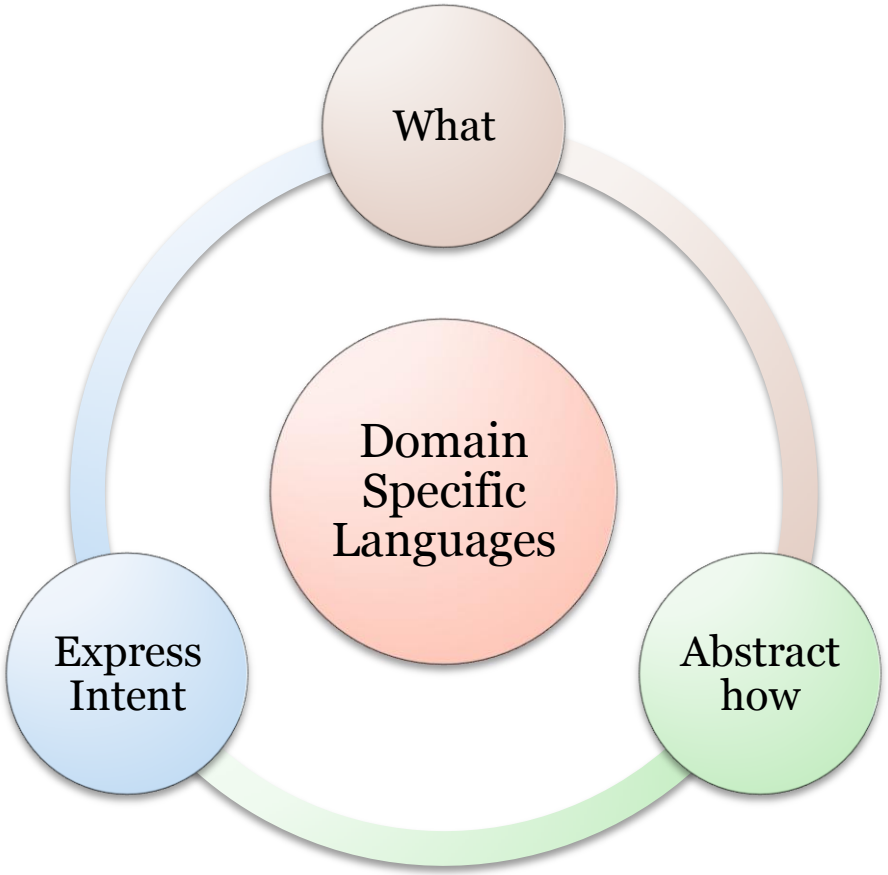
The problem…

- The difference between:

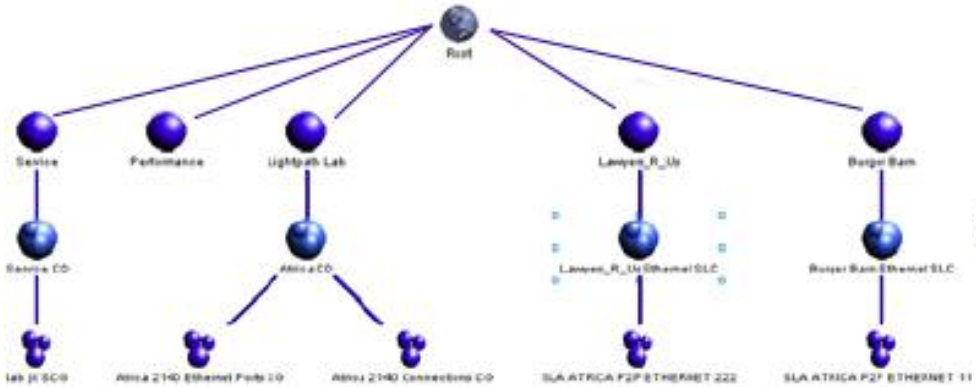


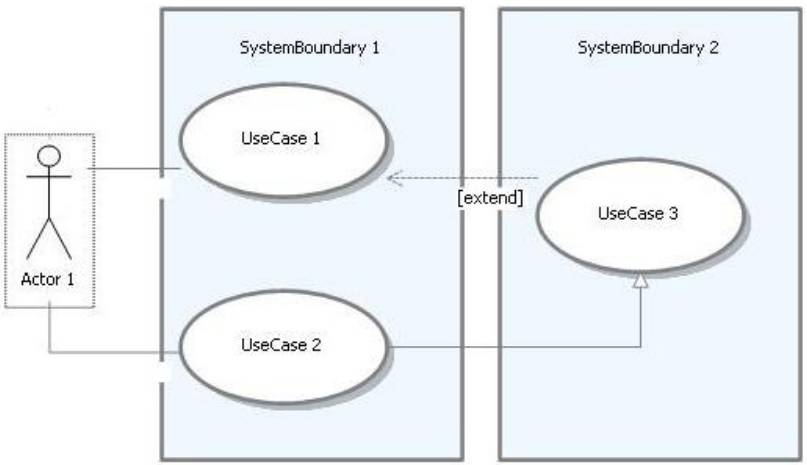
- #define do_what_I_meant_already

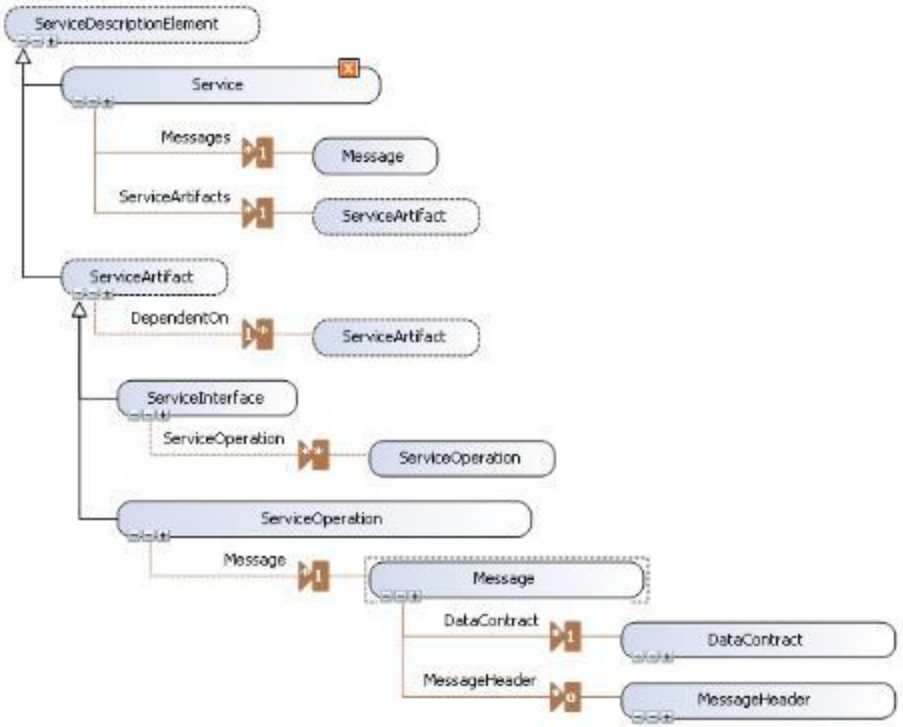
Domain Specific Languages



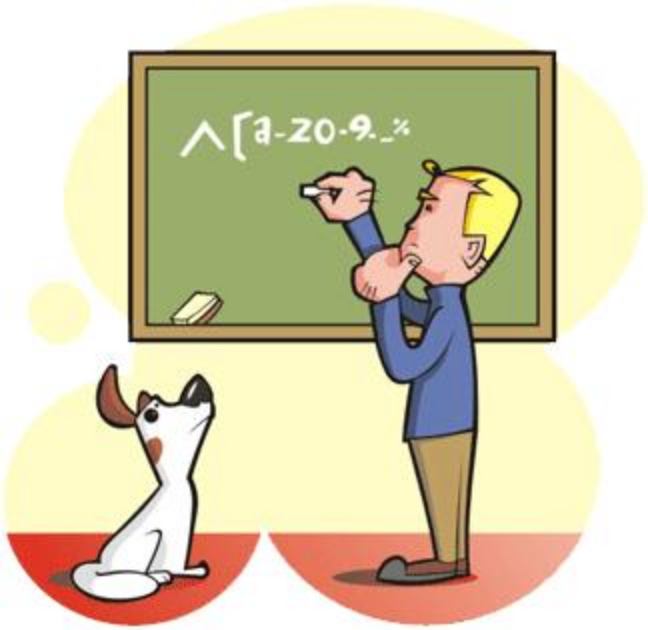
Graphical DSLs







External DSL




```
SELECT [Id]
      ,[Title]
      ,[Subtitle]
      ,[AllowsComments]
      ,[CreatedAt]
  FROM [Blogs]
```

Fluent Interface

^\d{3}-?\d{2}-?\d{4}$

```
Regex socialSecurityNumberCheck =
    Pattern.With.AtBeginning
        .Digit.Repeat.Exactly(3)
        .Literal("-").Repeat.Optional
        .Digit.Repeat.Exactly(2)
        .Literal("-").Repeat.Optional
        .Digit.Repeat.Exactly(4)
        .AtEnd;
```

Fluent Interface - Advance

```
User.FindAll(
        Where.User.Name == “Joe” &&
        Where.User.Birthday >= DateTime.Today &&
        Where.User.City.In(“London”, “Aarhus”)
);
```

Internal DSLs

- Based on existing language
  - Usually dynamic one
- Using the language facilities

- Common Languages:
  - **Boo**
  - Ruby
  - Python
  - Etc…

The Boo language

- Python derivative
  - white space agnostic mode
- Statically typed
  - optionally dynamic
- Runs on the CLR
- Open Compiler architecture
- Extremely malleable

White Space sensitive mode:

```
using session = sessionFactory.OpenSession():
   people = sess.Find("from Person")
   for p as Person in people:
         print "${p.Name} (${p.Initials})“
```

White Space agnostic mode:

```
using session = sessionFactory.OpenSession():
   people = sess.Find("from Person")
   for p as Person in people:
         print "${p.Name} (${p.Initials})“
   end
end
```

Building Boo DSL with C#

DSL Engine is language agnostic

Sample DSL: Business Processes

- Handling actions in ERP application

```
OnCreate Account:
  Entity.AccountNumber = date.Now.Ticks


OnCreate Order:
   if Entity.Total >Entity.Account.MaxOrderTotal:
     BeginManualApprovalFor Entity
```

What we already have?

- Domain classes (Order, Account)
- Compiler boot strapping:
  - Create & configure compiler
  - Compile file

Important syntactic sugar

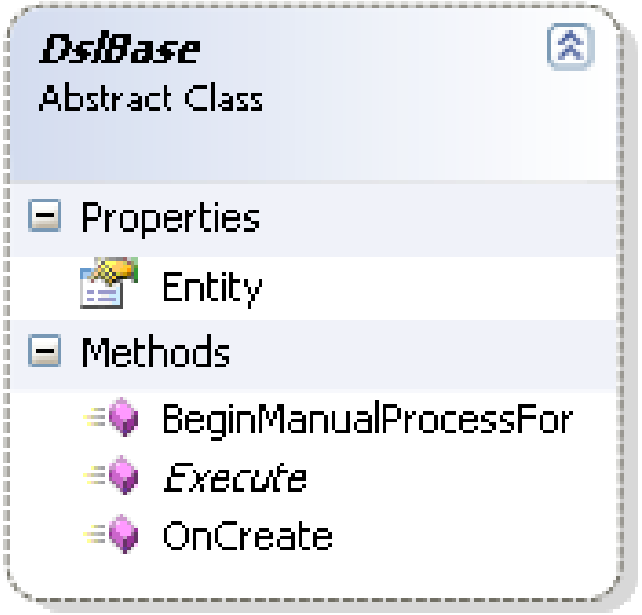
- ***Boo***

```
OnCreate Account:
   Entity.AccountNumber = date.Now.Ticks
```

- ***C#***

```
OnCreate(typeof(Account), delegate
{
    Entity.AccountNumber = DateTime.Now.Ticks;
});
```

Anonymous base class




```
OnCreate Account:
  Entity.AccountNumber = date.Now.Ticks

OnCreate Order:
  if Entity.Total >Entity.Account.MaxOrderTotal:
      BeginManualApprovalFor Entity
```

Demo

From:

```
OnCreate Account:
  Entity.AccountNumber = date.Now.Ticks

OnCreate Order:
   if Entity.Total > Entity.Account.MaxOrderTotal:
      BeginManualApprovalFor Entity
```

To:

```
class Sample(DslBase):
  override def Execute():
    accountAction = do:
      Entity.AccountNumber = date.Now.Ticks
    OnCreate(typeof(Account), accountAction)

    orderAction = do:
      if Entity.Total > Entity.Account.MaxOrderTotal:
          BeginManualApprovalFor(Entity)
    OnCreate(typeof(Order), orderAction)
```

Techniques

- Open Compiler
- Anonymous base class
- Context Parameters
- Extension Methods
- Meta Programming

Note: fairly new...

A bit about meta methods

```
import Boo.Lang.Compiler.Ast
import Boo.Lang.Compiler.MetaProgramming

[meta]
def assert(condition as Expression):
 return [|
   if not $condition:
    raise AssertionFailedException($(condition.ToCodeString()))
 |]

x as object = 1
assert x is null
```

Result:
Boo.Lang.Runtime.AssertionFailedException:
x is null

In the wild - Brail

```
My name is ${name}
<ul>
<%
   for element in list:
    output "<li>${element}</li>"
   end
%>
</ul>
```

In the wild – Rhino ETL

```
destination Final, Connection="Database":
  Command: """INSERT INTO OrdersWareHousing
      VALUES(@OrderID, @CompanyName,@ContactName)"""

pipeline OrdersWareHouse:
  CustomersFile >> CustomersAndOrders.Left
  OrdersFromDatabase >> CustomersAndOrders.Right
  CustomersAndOrders >> Final

 target default:
  Execute("OrdersWareHouse")
```

In the wild – Boo Build System

```
Task "init build dir”:
  MkDir("build")
  Cp(FileSet("lib/*.dll").Files, "build", true)

Task ”build boobs", ["build engine", "build extensions"]:
  bc = Booc(
       SourcesSet    : FileSet("tools/boobs/**/*.boo"),
       OutputFile    : "build/boobs.exe"
       )
  bc.ReferencesSet.Include("build/boobs.engine.dll")
  bc.ReferencesSet.Include("build/boo.lang.useful.dll")
  bc.Execute()
```

In the wild – Rhino Mocks Boo DSL

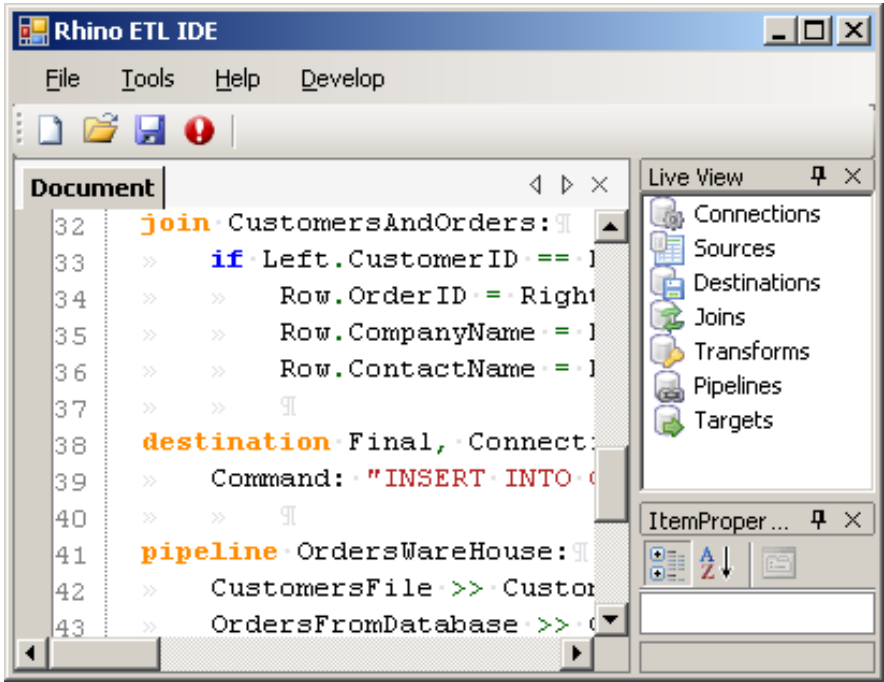
```
[Test]
def Get_data_objects_for_nonexistent_company_throws():
  with_mocks:
    database = mocks.CreateMock[of IDatabase]()
    userProvider = StubUserProvider("andrew", "bad corp")
    userProvider.MakeCurrent()

    assuming:
      database.GetUserID("andrew", "bad corp")
      returned 1
    assuming:
      database.GetCompanyID("bad corp")
      returned 0 # returning 0 == nonexistent company.

    execute:
      uds = UserDataService(database, userProvider)
      expect_throw FaultException[of GetDataObjectsFault]:
      uds.GetDataObjects()
```

In the wild - Specter

```
context "Empty stack":
    stack as Stack

    setup:
        stack = Stack()

    specify stack.Count.Must == 0

    specify "Stack must accept an item
  and count is then one":
        stack.Push(42)
        stack.Count.Must == 1
```

Tools

- Sharp Develop
- ICsharpCode.TextEditor



DSL Approaches

- Imperative DSL
  - Mostly library calls & syntax helpers
- Declarative DSL
  - Mostly to setup the underlying engine

Resources

- Boo website:
  http://boo.codehaus.org

- Boo Mailing List:
  http://groups-beta.google.com/group/boolang

- My blog:
  http://www.ayende.com/Blog/

Questions?