

# Dynamic Languages

Extending .NET Applications  
Fairfield/Westchester Code Camp  
2009-11-07

John C. Zablocki  
Development Lead, MagazineRadar, Inc.  
Adjunct, Fairfield University

# Agenda

Dynamic vs. Static Languages

Use Cases for Script Hosting

Dynamic Language Runtime (DLR)

DLR Hosting API

Tips and Techniques

Sample: IronPython, IronRuby, Boo

Questions

# Static vs. Dynamic Languages

## Static Languages

Type safety

Compile-time checking

Runtime performance

## Dynamic Languages

Late bound

Loosely typed

Highly expressive

# Script Hosting Use Cases

Why would you want to host a scripting language in your application?

Provide extensibility (the obvious answer)

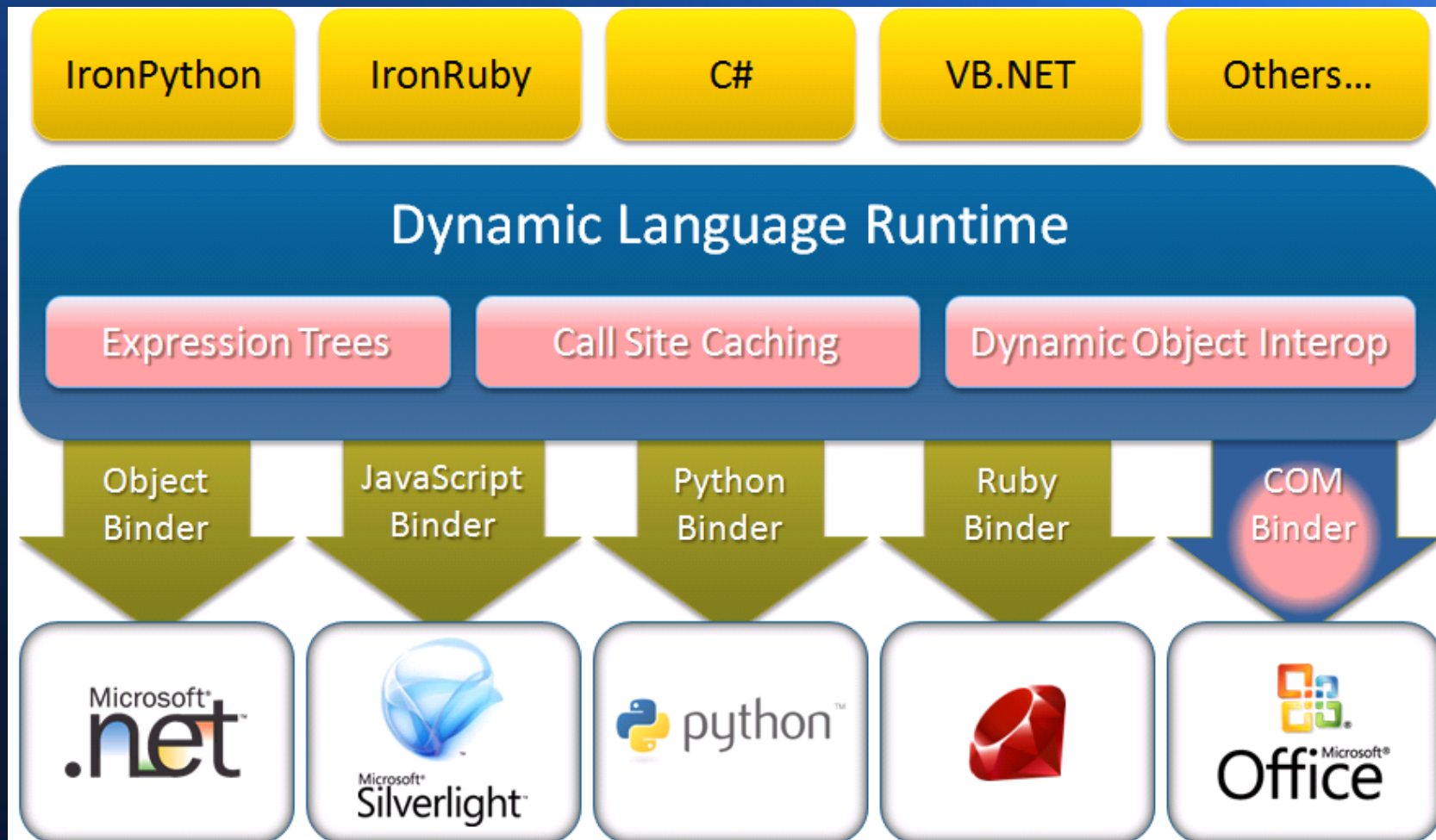
Replace complicated XML configuration (e.g. NAnt, EntLib, etc.) with meaningful code

Remove clutter from core application logic (validation, sanity checks, etc.)

Maintain transient logic outside of core application code (rules processing)

# Dynamic Language Runtime

(From CodePlex)



# Dynamic Language Runtime

Language implementation services

Language interoperability

Shared dynamic type system

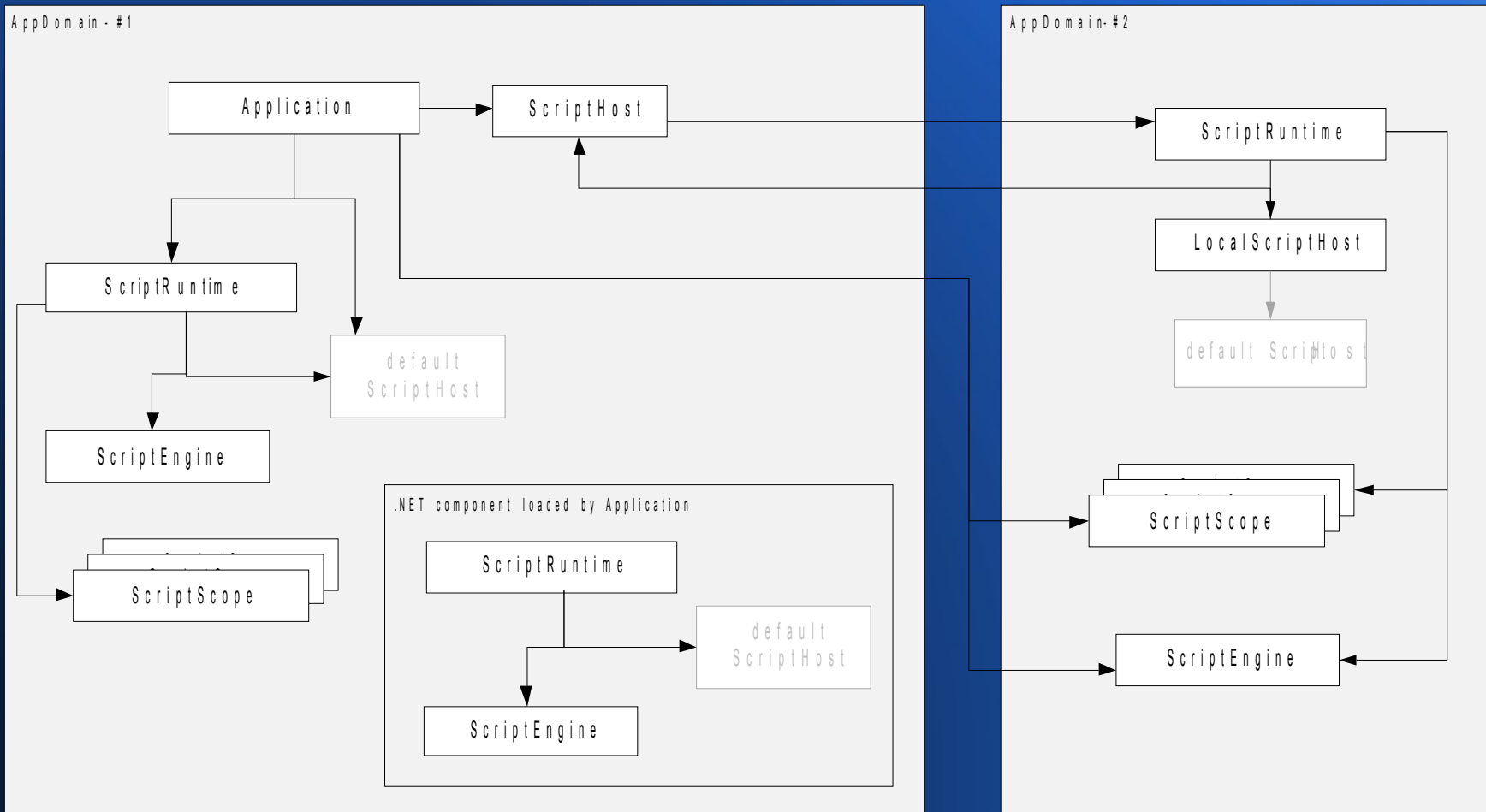
Runtime services

Fast dynamic code generation through various caching mechanisms

Utilities to allow statically typed objects to share in the dynamic message passing protocol

# Hosting API

(from CodePlex)



# Hosting API

ScriptRuntime

Starting point for hosting

Represents global script state (referenced assemblies, available engines, etc.)

Bound scopes (named globals)

Constructed with ScriptRuntimeSetup, which uses configuration settings

# Hosting API

ScriptEngine

Represents a DLR language implementation  
(PythonEngine, RubyEngine)

One engine, per-language, per-runtime

Methods to execute code and create ScriptScope  
instances

# Hosting API

ScriptScope

Essentially represents a namespace

Unit of isolation within a runtime

Has language affinity

Variables may be set and retrieved at scope

ScriptRuntime.Globals is actually an instance of a  
ScriptScope

# Hosting API

ScriptSource

Represents source code

Provides means for code execution and compilation

Created from ScriptEngine or ScriptScope instances

# Tips and Techniques

- The script is an extension of the application, not core application logic
- Treat the script as if it were a plugin or provider implementing an interface.
- Create a simple object graph to act as the controller between the script and model
- Provide the script with function references to work indirectly with model objects

# The IronPython Language

CLI implementation of the Python language

Hosted on CodePlex

Integration with many common Python libraries

Built on top of the DLR

Sample: Hosted IronPython

# The IronRuby Language

CLI implementation of the Ruby language

Hosted on RubyForge

Built on top of the DLR

Sample: Hosted IronRuby

# The Boo Language

Object oriented, statically typed language written for the CLI

Has a Python-like feel (whitespace, blocks)

Supports Duck typing (dynamic-like behavior)

Flexible syntax, extensible compiler and InteractiveInterpreter make it strong choice for DSL implementations

Sample: Hosted Boo

# Resources

<http://www.codevoyeur.com> – DLR Samples, other OSS samples, presentations and articles

<http://www.dllhell.net> – My blog

<http://www.codeplex.com/dlr>

<http://www.codeplex.com/IronPython>

<http://boo.codehaus.org>

<http://www.ironruby.net>

# Questions?