F# 3.0 Information Rich Programming

Don Syme, Principal Researcher, Microsoft Research
The F# Team, Microsoft Visual Studio

http://fsharp.net
Today’s talk is very simple
Proposition 1
The world is information-rich
Proposition 2
Modern applications are information-rich
Proposition 3
Our languages are information-sparse
Proposition 4
This is a problem
The future of F# is about fixing this
To fix it we will use magic 😊
The magic we’re adding to F# is called **Type Providers**
Type Providers + LINQ = Language Integrated Data and Services
A Coding Challenge:

The Chemical Elements, Strongly Typed
Language Integrated Web Data
The Magic: Type Providers
How would we do this in strongly typed languages today?
let Query<'T>(query:string) : 'T =
let query = query.Replace("\","")
let queryUrl = sprintf "http://api.freebase.com/api/service/mqlread?query=%s" query
let request : HttpWebRequest = downcast WebRequest.Create(queryUrl)
request.Method <- "GET"
request.ContentType <- "application/x-www-form-urlencoded"
let response = request.GetResponse()
let result = try
    use reader = new StreamReader(response.GetResponseStream())
    reader.ReadToEnd();
finally
    response.Close()
let data = Encoding.Unicode.GetBytes(result);
let stream = new MemoryStream()
stream.Write(data, 0, data.Length);
stream.Position <- 0L
let ser = JsonDataContractJsonSerializer(typeof<Result<'T>>)
let result = ser.ReadObject(stream) :?> Result<'T>
if result.Code <> "http://api/status/ok" then
    raise (InvalidOperationException(result.Message))
else
    result.Result

let elements = Query<ChemicalElement> array("[\{"type":'/chemistry/chemical_element',"name":null,"boiling_point":null,"atomic_mass":null\}]")

elements |> Array.iter(fun element->printfn "%A" element)
The Magic: Type Providers

Type Providers

- Data and services at your fingertips
- Scalable (‘000,000s of types)
- Navigation, Intellisense
- Integrate with LINQ queries
- No code gen
A Type Provider is....

A design-time component that provides a computed space of types and methods...

A compiler/IDE extension...

An adaptor between data/services and .NET languages...
public interface ITypeProvider
{
    Type GetType(string name, BindingFlags bindingAttr);

    Expression GetInvokerExpression(
        MethodBase syntheticMethodBase,
        ParameterExpression[] parameters);

    event System.EventHandler Invalidate;

    Type[] GetTypes();
}
let Query<'T>(query:string) : 'T =
    let query = query.Replace("","\")
    let queryUrl = sprintf "http://api.freebase.com/api/service/mqlread?query=%s" query
    let request : HttpWebRequest = downcast WebRequest.Create(queryUrl)
    request.Method <- "GET"
    request.ContentType <- "application/x-www-form-urlencoded"
    let response = request.GetResponse()
    let stream = new MemoryStream()
    stream.Write(data, 0, data.Length)
    stream.Position <- 0L
    let ser = JsonDataContractJsonSerializer(typeof<Result<'T>>)
    let result = ser.ReadObject(stream) :?> Result<'T>
    if result.Code <> "\api\status\ok" then
        raise (InvalidOperationException(result.Message))
    else
        result.Result

let elements = Query<ChemicalElement> array("[{'type':'chemistry/chemical_element','name':null,'boiling_point':null,'atomic_mass':null}]")
elements |> Array.iter(fun element->printfn "%A" element)
How do we mediate today? Codegen, Codegen, Codegen!!

Ok, but… there are problems 😊

UI designs
WCF services
Resources
OData
Databases
...
Note: F# still contains no data

Open architecture

You can write your own type provider
One stop shop for Data.

Get all data you need for your insights: trusted commercial and premium public domain data.

Interact with your data spatially.
1 - 10 Results of 77

Sort By: Date Added Name Publisher

1. DATA.GOV
   2006 - 2008 Crime in the United States
   published by Government of the United States of America
   Extraction of offense, arrest, and clearance data as well as law enforcement staffing information from the FBI's Uniform Crime Reporting (UCR) Program.

2. esri
   2010 Key US Demographics by ZIP Code, Place and County - Esri
   published by esri
   Esri 2010 Key US Demographics by ZIP Code, Place and County Data is a select offering of the demographic data required to understand a market. Esri 2010 Key Demographics contains current-year updates for...
Language Integrated Data Market Directory
type Netflix = ODataService<"http://odata.netflix.com">
type SQL = SqlConnection("Server=\SQLEXPRESS\... ")
type EmeaSite = SharePointSite<"http://myemea/">
let avatarTitles =
    query { for t in netflix.Titles do
        where (t.Name.Contains "Avatar") }
let avatarTitles =
    query {
        for t in netflix.Titles do
            where (t.Name.Contains "Avatar")
            sortBy t.Name
            take 100
    }
TypeProviders: The Vision

- ...web data
- ...data markets
- ...WMI & active directory
- ...a spreadsheet
- ...web services
- ...CRM data
- ...social data
- ...SQL data
- ...XML data
- ...

- strongly typed
- without explicit codegen
- extensible, open
Summary

The world is information rich

Our languages need to be information-rich too

The F# Future? Consume it! Directly! Strongly typed! No walls!
Latest Books about F#

Visit www.fsharp.net
Professional Tools
**In Summary**

<table>
<thead>
<tr>
<th>Simple, expressive, productive addition to .NET</th>
<th>Ready for supported use in VS2010</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple code, Simple parallelism</td>
<td>An amazing data-rich future ahead</td>
</tr>
</tbody>
</table>

F#
Questions

http://fsharp.net
How would we do this in strongly typed languages today?
public interface ITypeProvider
{
    Type GetType(string name,
                  BindingFlags bindingAttr);

    Expression GetInvokerExpression(
                                      MethodBase syntheticMethodBase,
                                      ParameterExpression[] parameters);

    event System.EventHandler Invalidate;

    Type[] GetTypes();
}
1 - 10 Results of 77

Sort By: Date Added Name Publisher

DATA.GOV 2006 - 2008 Crime in the United States
published by: Government of the United States of America
date added: SEP 30, 2010

Extraction of offense, arrest, and clearance data as well as law enforcement staffing information from the FBI's Uniform Crime Reporting (UCR) Program.

esri 2010 Key US Demographics by ZIP Code, Place and County - Esri
published by: esri
date added: OCT 26, 2010

Esri 2010 Key US Demographics by ZIP Code, Place and County Data is a select offering of the demographic data required to understand a market. "Esri 2010 Key Demographics" contains current year updates for...
How do we mediate today?

WebRequest
XmlReader
ReadJSON
SqlReader
...

Untyped, non-navigable

UI designs
WCF services
Resources
Web services
Databases

WebRequest
XmlReader
ReadJSON
SqlReader
...

objects
strings
bytes

program
How do we mediate today? Codegen, Codegen, Codegen!!

Ok, but... there are problems 😊

UI designs → xaml designer
WCF services → svcutil.exe
Resources → resgen.exe
OData → datasvcutil.exe
Databases → edmgen.exe
... → yacg.exe

types + code → program
Implementation

FSC.EXE

FSharp.LanguageService.dll

FSI.EXE

System.Type (synthetic)

app.exe or library.dll

Some.Unknown.Identifier

-provider.dll

-how to compile method calls to synthetic types

Provider.Runtime.dll
Example #6: Xbox Path of Go

...F# rocks - building out various algorithms for DNA processing here and it's like a drug. 12-15 at Amyris use F#

F# has been phenomenally useful. I would be writing a lot of this in Python otherwise and F# is more robust, 20x - 100x faster to run and faster to develop.

Amyris BioTechnologies
let avatarTitles =
    query { for t in netflix.Titles do
      where (t.Name.Contains "Avatar")
      select t }

let avatarTitles =
query { for t in netflix.Titles do
    where (t.Name.Contains "Avatar")
yield t }
let avatarTitles =
    query { for t in netflix.Titles do
             where (t.Name.Contains "Avatar")
             sortBy t.Name  }
let avatarTitles =
query { for t in netflix.Titles do where (t.Name.Contains "Avatar") sortBy t.Name take 100 }
<table>
<thead>
<tr>
<th><strong>query</strong> { ... }</th>
<th><strong>seq</strong> { ... }</th>
</tr>
</thead>
<tbody>
<tr>
<td>for, where, if-then, yield, yield!</td>
<td>for, where, if-then, yield, yield!</td>
</tr>
<tr>
<td>sortBy sortByDescending</td>
<td>match, if-then-else, try/with, try/finally</td>
</tr>
<tr>
<td>distinct, head, last, single, nth</td>
<td></td>
</tr>
<tr>
<td>skip, skipWhile, take, takeWhile</td>
<td></td>
</tr>
<tr>
<td>groupBy, join, groupJoin</td>
<td></td>
</tr>
<tr>
<td>averageBy, sumBy, maxValBy, minValBy</td>
<td></td>
</tr>
<tr>
<td>forall, exists, contains, find</td>
<td></td>
</tr>
<tr>
<td>headOrDefault, lastOrDefault singleOrDefault</td>
<td></td>
</tr>
<tr>
<td>State machines for non-IQueryable LINQ compiled for IQueryable</td>
<td>State machine compiled</td>
</tr>
</tbody>
</table>
Example #2 (power company)

**Interoperation** ... Seamless. The C# programmer need never know.

**Units of measure** ... a huge time saver...it eradicates a whole class of errors...

**Exploratory programming**... Working with F# Interactive allowed me to explore the solution space more effectively.

**Unit testing** ...a joy to test. There are no complex time-dependent interactions to screw things up....

**Parallelism**... The functional purity ... makes it ripe for exploiting the inherent parallelism in processing vectors of data.

**Code reduction**... ... vectors and matrices...higher order functions eat these for breakfast with minimal fuss, minimal code. Beautiful.

**Lack of bugs**... Functional programming can feel strange. .. once the type checker is satisfied that’s often it, it works.
Implementation (generative)

FSC.EXE

Some.Unknown.Identifier + resolution context

System.Type (real)

app.exe or library.dll

types

-r:Provider.dll

edmgen.exe

Provider.Runtime.dll
type Expr =
| True
| And of Expr * Expr
| Nand of Expr * Expr
| Or of Expr * Expr
| Xor of Expr * Expr
| Not of Expr

class True Expr

class And : BinExpr

class Nand : BinExpr
type Event =
| Price of float
| Split of float
| Dividend of float<money>

public abstract class Event {
}
public abstract class PriceEvent : Event
{
    public Price Price { get; private set; }
    public PriceEvent(Price price)
    {
        this.Price = price;
    }
}

public abstract class SplitExpr : Event
{
    public double Factor { get; private set; }
    public SplitExpr(double factor)
    {
        this.Factor = factor;
    }
}

public class DividendEvent : Event {
}
A Research/Product Partnership

Microsoft Visual Studio

.NET Generics
Comega/LINQ
F#/.NET/C#/VB Async

Microsoft Research

Industry/Community
Research
Product Team
F# in the Enterprise

Accelerated Analytical and Parallel .NET Development with F# 2.0

Version 0.4 – February 28, 2011

Abstract

Microsoft F# 2.0 is a flexible and efficient language for developing analytical, data-rich, computational and parallel software components. Backed by a professional toolset in Visual Studio 2010 and incorporating the full power of the .NET platform, the language is a perfect companion to existing professional development techniques for enterprise computing.

F# 2.0 combines functional and object-oriented programming to enable the rapid creation of simple code to solve complex problems. It is well suited for performing heavy numeric computations across large data sets and has been successfully applied in financial, statistical, parallel, scientific, engineering, testing, event-processing, tool-development, and general-purpose software components.

This paper outlines the benefits of F# for enterprise scenarios, describes case studies and other real-world experiences with F#, and includes information on finding and developing F# programming skills for your organization.

Contents
I've been coding in F# lately, for a production task.

F# allows you to move smoothly in your programming style... I start with pure functional code, shift slightly towards an object-oriented style, and in production code, I sometimes have to do some imperative programming.

I can start with a pure idea, and still finish my project with realistic code. You're never disappointed in any phase of the project!