The Future of F#: data and information at your fingertips

Joe Pamer
Senior Development Lead, F#
Microsoft

Today we're going to discuss the present and future of F#

What is F#?

all-vs2010 - Microsoft Visual Studio

File Edit View Project Build Debug Team Data Tools Architecture Test Analyze Window Help

Debug | Any CPU | New Work Item

ilwrite.fs | ilxgen.fs | fsc.fs | fscopts.fs

Server Explorer | Toolbox

```
let monoCompilerSvc = "Mono.CompilerServices.SymbolWriter, Version=2.0.0.0, Culture=neut
let ctor (asmName:string) (clsName:string) (args:obj[]) =
    let asm = Assembly.Load(asmName)
    let ty = asm.GetType(clsName)
    System.Activator.CreateInstance(ty, args)

let createSourceMethodImpl (name:string) (token:int) (namespaceID:int) =
    ctor monoCompilerSvc "Mono.CompilerSe

let createWriter (f:string) =
    ctor monoCompilerSvc "Mono.CompilerSe

//-----------------------------------------
// MDB Writer.  Generate debug symbols us
//-----------------------------------------
```

100 %

Ready | Ln 3

```
let PrintColors(greyScale) =
    // A completely ad-hoc algorithm for turning a "semantic depth" into pretty alternating red/blue
    // progressively-darkening colors.  In order to emphasize contrast at the border of adjacent regions,
    // the left edge uses a thin gradient of a darker color.
    let MAX = 240uy
    let start = MAX-20uy
    let reduce =    if greyScale then [| for i in 0uy .. 9uy -> 6uy*(i/2uy) |]
                    else [| for i in 0uy .. 9uy -> 8uy*(i/2uy) |]
    for depth in 0..reduce.Length-1 do
        let depth = depth % reduce.Length
        let x = start - reduce.[depth]
        let g = start - 0uy - reduce.[depth]/2uy  // green has lots of luminance, and small changes can appear abrupt
        let z = x - 40uy
        if (depth % 2 = 0) then
            let x,g,z,MAX,zMAX =
                if greyScale then
                    let x = x + 10uy
                    x, x, x - 40uy, x, x - 40uy
                else
                    x, g, z, MAX, MAX
            printfn "%d,%d,%d,%d,%d,%d" zMAX z z MAX g x
        else
            let x,g,z,MAX,zMAX =
                if greyScale then
                    let x = x - 10uy
                    x, x, x - 40uy, x, x - 40uy
                else
                    x, g, z, MAX, MAX
            printfn "%d,%d,%d,%d,%d,%d" z z zMAX x g MAX
```

F# is…

…a productive, supported, interoperable, functional language that allows you to write simple code to solve complex problems

F# is…

- A functional/object-oriented programming language on .NET
- Fully-supported by Microsoft

F# 2.0 in context

Modern application development is generally composed of three tasks:

- Data, Services, Information Access
- Transformation, Analysis, Parallelism
- Presentation, Publication

F# can be used for just about anything, but it currently excels at analytical programming

F# Today: F# 2.0

Goals for F# 2.0

- Succinct, Expressive, Functional language
  - Productive, simple, powerful, and fun language
- Excel at Parallel, Explorative, Data-rich, Algorithmic programming tasks
  - Extends the .NET platform to important new audiences
- Be Innovative, Platform-Leading, 1st-class .NET language
  - Asynchronous workflows, units-of-measure, VS tooling
- Non-goal: Be a replacement for C#/VB/C++
  - Augments and builds on .NET as multi-language platform

F# Today: F# 2.0

- Compiler and runtime source code available at fsharppowerpack.codeplex.com
  - Apache 2.0 License

F# PowerPack, with F# Compiler Source Drops

CodePlex
Open Source Community

Search all CodePlex projects | Search

Home | Downloads | Discussions | Issue Tracker | Source Code | People | License | RSS

View All Comments | Print View | Page Info | Change History (all pages)

Search Wiki & Documentation

Home

The F# PowerPack, with F# Compiler Source Drops

The F# PowerPack is a collection of libraries and tools for use with the F# programming language provided by the F# team at Microsoft.
The F# PowerPack includes open source code drop(s) of the F# compiler and core library under the OSS approved Apache 2.0 license, but does not include binary builds of these.

If you're looking to get, learn, install or use F#, go to www.fsharp.net

The additional libraries in the F# PowerPack are functionality which is not part of the core F# release, but enables some development scenarios with F#. The PowerPack include features such as a basic Matrix library and supporting math types, FsLex and FsYacc tools for lexing and parsing, support for using F# with LINQ-based libraries, and a tool for generating HTML documentation from F# libraries. This functionality, which has previously been available in the F# CTP releases, is now available on CodePlex. The F# PowerPack will continue to evolve seperately from the main F# releases, and the features will continue to be improved and iterated upon.

Sub Pages: FsLex FsYacc.

Getting Started

- Download the latest release
- Download the F# compiler and tools
- Browse the source code

Feedback

Download

CURRENT	May 2010
DATE	Thu May 20 2010 at 3:00 AM
STATUS	Beta
RATING	No Ratings 5569 downloads
MORE	View all downloads

Activity 7 30 All days

Page Views	1534
Visits	613
Downloads	310
Application Runs	N/A

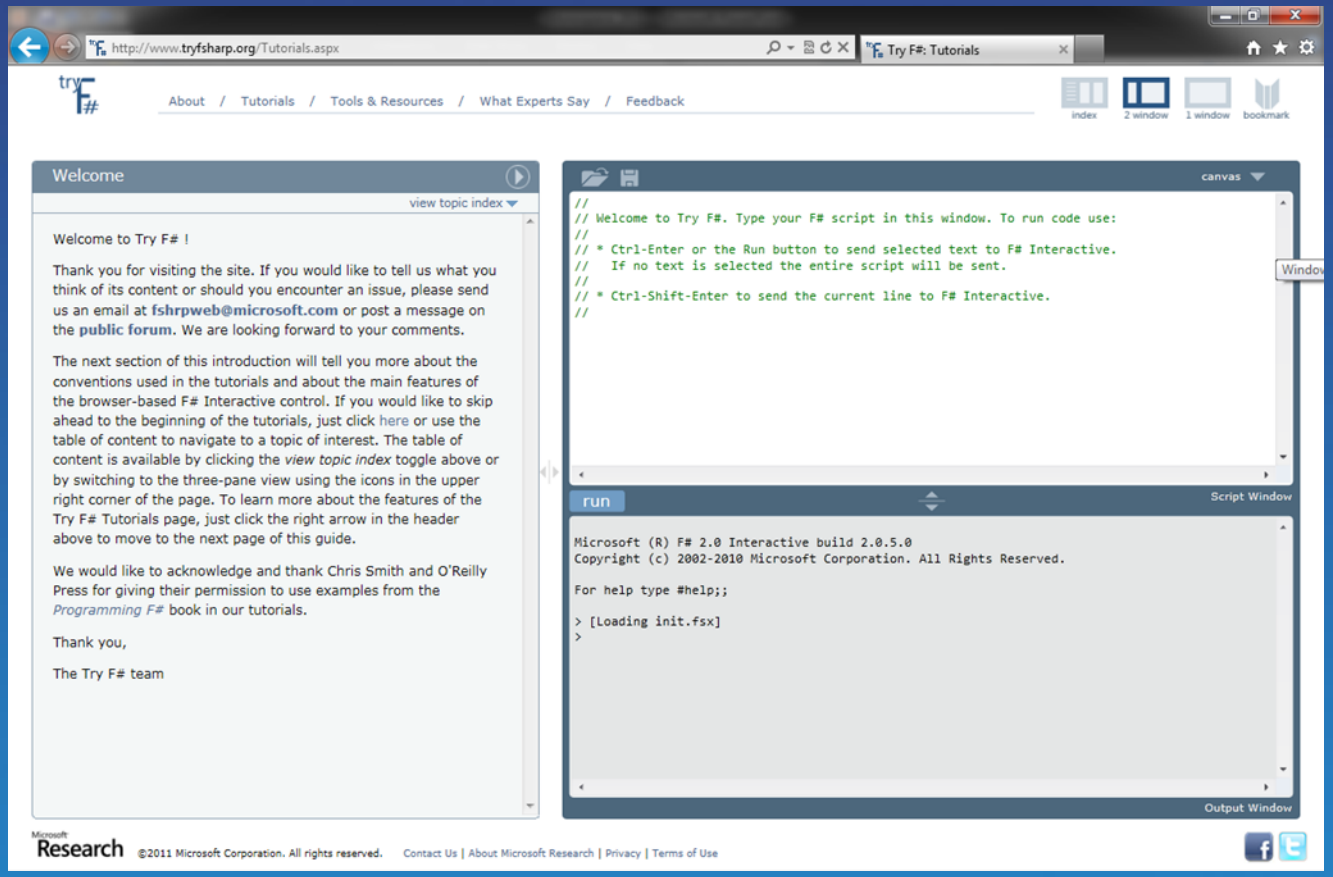
View Detailed Stats

Related Projects

- F# cross-platform packages and samples

F# Today: F# 2.0

Download it, or try it now at tryfsharp.org



F# Tomorrow

Where are we taking the language?

Understanding F# 3.0 is very simple…

Two propositions

Proposition 1

The world is information-rich

Proposition 2

Our languages are information-sparse

This is a problem…

- Especially for statically-typed languages – often an impedance mismatch
  - Manual integration of external tools into your build process
  - Code/IL generation
  - Loss of type information via up-casting or… *strings*

Today, the more information-rich the environment, the greater the disadvantage for statically-typed languages

That shouldn't be the case

- Data sources often have rich schemas and associated data definitions
- Static definitions of data should make your experience better, not worse!

F# 3.0 is about language and tooling foundations to address these problems

F# 3.0 in context

- Data, Services, Information Access
- Transformation, Analysis, Parallelism
- Presentation, Publication

What can we do?

Challenge some of our assumptions

- Compilers
- Tooling
- Language architecture

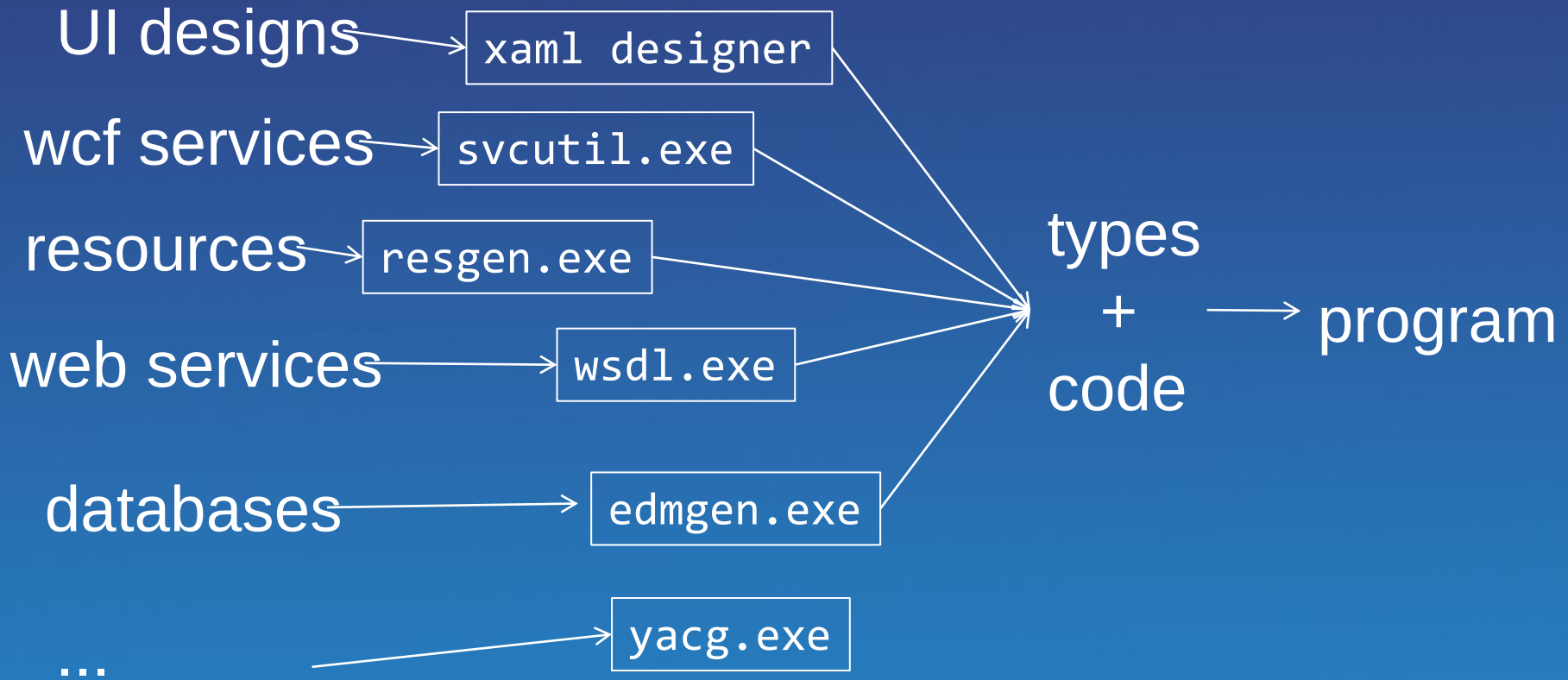
Challenge our notion of libraries

- Information spaces are really just libraries we can use as part of our ambient environment
  - Similar to how the .NET framework is part of the ambient environment of .NET languages

A Demonstration…

Task: Explore various categorizations of programming languages available online

How do we mediate today? Where do the types come from?



```
// Freebase.fsx
// Example of reading from freebase.com in F#
// by Jomo Fisher
#r "System.Runtime.Serial
#r "System.ServiceModel.W
#r "System.Web"
#r "System.Xml"

open System
open System.IO
open System.Net
open System.Text
open System.Web
open System.Security.Auth
open System.Runtime.Seria

[<DataContract>]
type Result<'TResult> = {
    [<field: DataMember(N
    Code:string
    [<field: DataMember(N
    Result:'TResult
    [<field: DataMember(N
    Message:string
    }

[<DataContract>]
type ChemicalElement = {
    [<field: DataMember(N
    Name:string
    [<field: DataMember(N
    BoilingPoint:string
    [<field: DataMember(N
    AtomicMass:string
    }
```

```
let Query<'T>(query:string) : 'T =
    let query = query.Replace("'","\"")
    let queryUrl = sprintf "http://api.freebase.com/api/service/mqlread?query=%s"
"{\"query\":"+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 = Json.DataContractJsonSerializer(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 are we addressing this in F# 3.0?

Exploring web data

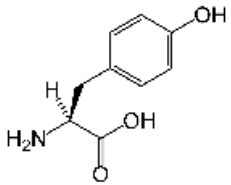
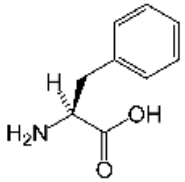
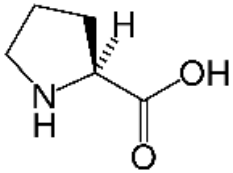
demo

Our principle goal for F# 3.0 is to build language foundations for strongly typed access to external named data and services

- Data and Services at Your Fingertips
- Scalable (millions of types)
- Navigable - Intellisense!
- Rich with metadata & assists (XML docs, “go to definition”)
- Integrates with LINQ queries
- Integrates with the REPL (or not)
- No manual code gen, often no code gen at all

Type Providers

Data Grid

Name	MainImage	Blurb	DNACodons	Id
Tyrosine		[Tyrosine (abbreviate Aside from being a pr A tyrosine residue als	[TAC; TAT]	/en/tyrosine
Phenylalanine		[Phenylalanine (abbr Phenylalanine is foun L-Phenylalanine is bic	[TTC; TTT]	/en/phenylalanine
Proline		[Proline (abbreviated Proline is biosyntheti The distinctive cyclic	[CCT; CCG; CCC; ...]	/en/proline
Cysteine		[Cysteine (abbreviate	[TGC; TGT]	/en/cysteine

A Type Provider is….

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

A compiler/IDE extension

The static counterpart to dynamic languages

Intellisense for data

Extensible and open

Breaking down walls!

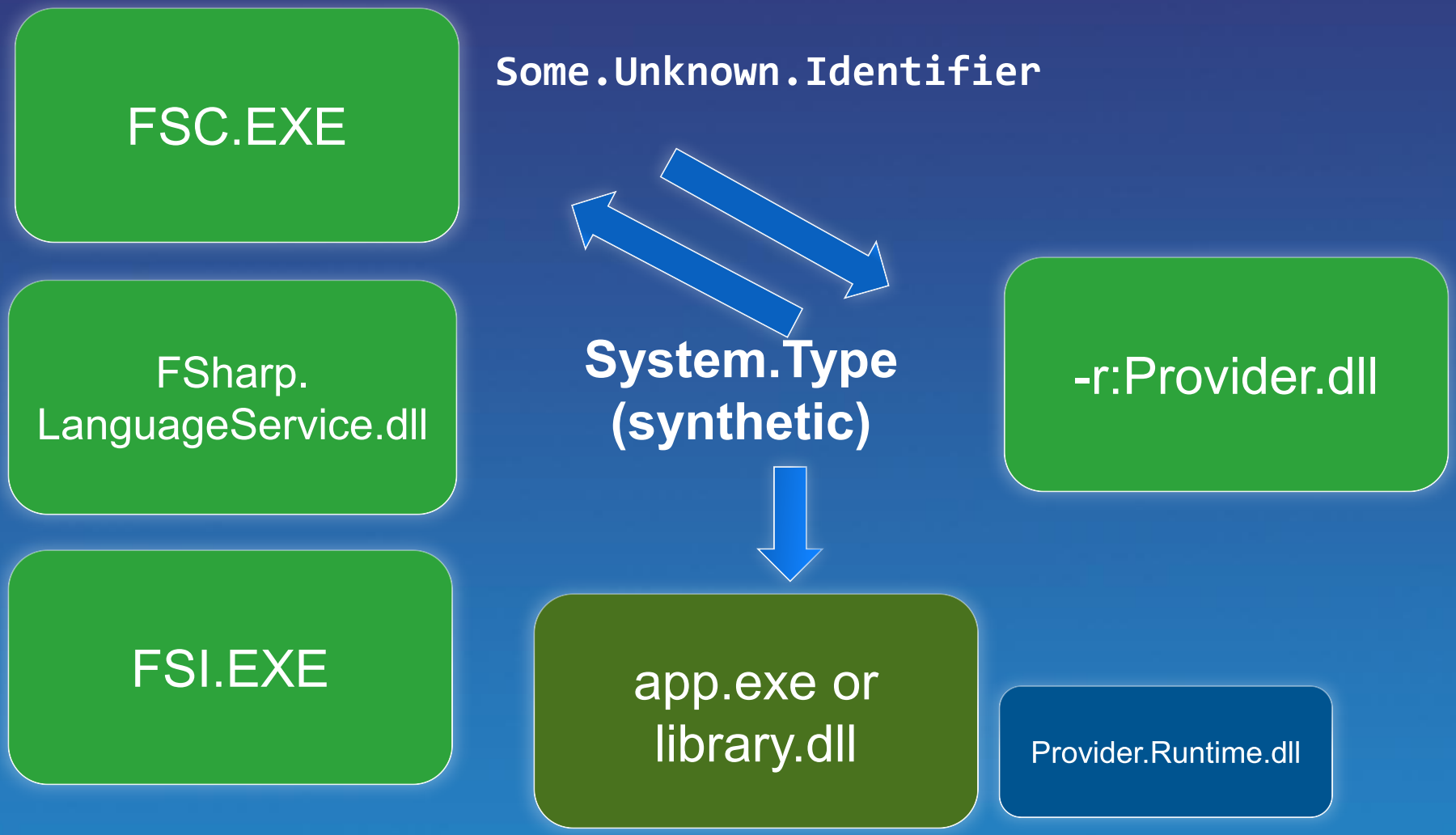
Concretely…

Extensible name resolution via "Type Providers"

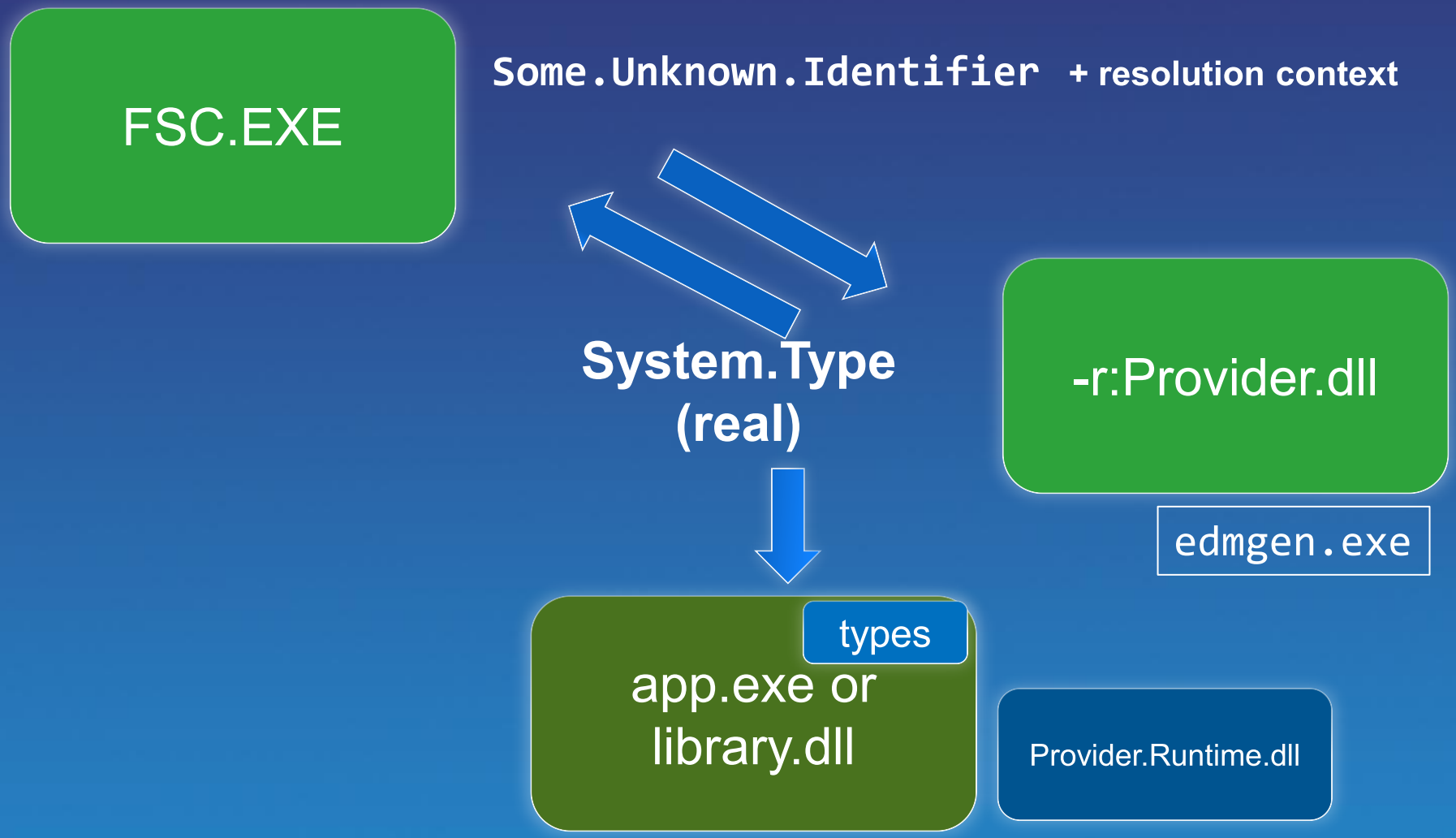
- Type Providers are DLLs
- Map information sources into the .NET type system
- Based on schemas (or sample information)
- Generative and synthetic ("erasure-based") options

Synthetic and Generative Providers

Implementation (synthetic)



Implementation (generative)



Type Providers and “big data”

Display crime statistics for cities in Missouri

Windows Azure Marketplace DataMarket - Browse Datasets - Windows Internet Explorer

https://datamarket.azure.com/browse

Favorites | Windows Azure Marketplace DataMarket - Brows... | Page | Safety | Tools

Windows Azure Marketplace DataMarket

Sign In | Register

Learn | Browse | Partners | My Data | Help | Search the Marketplace

1 - 10 Results of 77

1 2 3 4 ... 8 next

CATEGORIES

- Entertainment and Media
- Financial
- Health and Wellness
- Location Based Services
- Mathematics
- Movies and Events
- News
- Points of Interest
- Real Estate
- Retail and Merchandise

Sort By: Date Added | Name | Publisher

DATA.GOV

2006 - 2008 Crime in the United States

date added: SEP 30, 2010

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.

esri

2010 Key US Demographics by ZIP Code, Place and County - Esri

date added: OCT 26, 2010

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

Internet | Protected Mode: On | 100%

Language Integrated Data Market Directory

demo

Note: No code spew!

Open architecture

You can write your own type provider

The API for Type Providers

```
public interface ITypeProvider
    {
        Type GetType(string name);

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

        event System.EventHandler Invalidate;

        Type[] GetTypes();
    }
```

A type provider (one type, one static property)

```
type SampleTypeProvider(config: TypeProviderConfig) as this =

    inherit TypeProviderForNamespaces()

    let oneType = ProvidedTypeDefinition("Samples.TypeSpace","OneType",Some typeof<obj>)
    let prop = ProvidedProperty( "Name",
                                 typeof<string>,
                                 GetterCode= (fun args -> <@@ “Hello" @@>))
    oneType.AddMember prop

    do this.AddNamespace("Samples.TypeSpace", [ oneType ])

[<assembly:TypeProviderAssembly>]
do()
```

What providers are we distributing with F# 3.0?

SQL Entities / Linq2SQL

```
type Data =
      SqlDataConnection<"Server='.\\SQLEXPRESS'..">

let db = SQL.GetDataContext()

let data = query {for item in db.Customers do
                      select item.Name}
```

OData

```
type Netflix = ODataService<"http://odata.netflix.com">

let db = Netflix.GetDataContext()

let data = query {for title in db.Title do
                      select title.Name}
```

WSDL

```
type Data = WsdlService<"http://www.foosvcs.com/WSDL">

let financials = Data.GetServiceContext()

financials.GetQuotes "IBM"
```

Code focused!

```
[<Generate>]
type dbSchema  = Microsoft.FSharp.Data.TypeProviders.SqlDataConnection<"Data Source=localhost;Initial Catalog=FSharpSamp
let db = dbSchema.GetDataContext()
let myQuery = query {
    for item in db.st
~
```

- Connection
- Course
- CourseSelection
- DataContext
- Student

property dbSchema.ServiceTypes.SimpleDataContextTypes.FSharpSample.Student: System.Data.Linq.Table< dbSchema.ServiceTypes.Student>

Gets the 'Student' entities from the SQL connection. This property may be used as the source in a query expression.

TypeProviders: The Vision

- …web data
- …data markets
- …WMI & active directory
- …spreadsheets
- …web services
- …CRM data
- …social data
- …SQL data
- …XML data
- …

strongly typed

without explicit codegen

extensible, open

The mission

- F# 2.0 is an accelerator for solving computationally complex problems
- F# 3.0 is an accelerator for solving data complex problems

Summary

The world is information rich

Our languages need to be information-rich too

F# 3.0? Consume it! Directly! Strongly typed! No walls!

F# 3.0 is available today*

```
[<Generate>]
type dbSchema  = Microsoft.FSharp.Data.TypeProviders.SqlDataConnection<"Data Source=localhost;Initial Catalog=FSharpSamp
let db = dbSchema.GetDataContext()
let myQuery = query {
    for item in db.st
```

Connection
Course
CourseSelection
DataContext
Student

property dbSchema.ServiceTypes.SimpleDataContextTypes.FSharpSample.Student: System.Data.Linq.Table<dbSchema.ServiceTypes.Student>

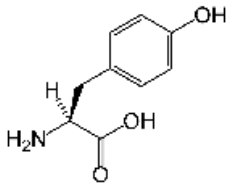
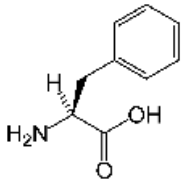
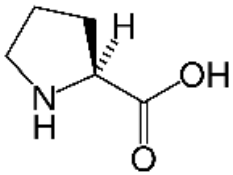
Gets the 'Student' entities from the SQL connection. This property may be used as the source in a query expression.

Check out the Visual Studio 11 Developer Preview:
http://msdn.microsoft.com/en-US/vstudio/hh127353

*In Alpha form!

Questions?

Data Grid

Name	MainImage	Blurb	DNACodons	Id
Tyrosine		[Tyrosine (abbreviate Aside from being a pr A tyrosine residue als	[TAC; TAT]	/en/tyrosine
Phenylalanine		[Phenylalanine (abbr Phenylalanine is foun L-Phenylalanine is bic	[TTC; TTT]	/en/phenylalanine
Proline		[Proline (abbreviated Proline is biosyntheti The distinctive cyclic	[CCT; CCG; CCC; ...]	/en/proline
Cysteine		[Cysteine (abbreviate	[TGC; TGT]	/en/cysteine

http://fsharp.net

Thanks!