Title: F# Type Providers – Unleashing the Semantic Web with Programming Languages

Abstract: The Semantic Web and the Linking Open Data initiative address the need for machine-readable Web content where the Web is no longer seen as mainly a web of documents, but rather a web of objects or entities with relations between them. And yet, it does take a certain amount of effort to create these entities and relations at web scale. We will show how programming languages can help with the democratization of the authoring task.

The worlds of programming languages and the semantics web seem universes apart. However, in this talk we will show how a simple and intuitive change in programming language architecture can lead to a wonderful union between these two worlds, combining the simplicity and power of modern professional programming tools with the masses of organized data appearing through the structured, organized, schematized data sources now populating the web.

In particular, the talk will focus on the following aspects:

- F# type providers and their applications to strongly typed programming with web ontologies and other rich structured information sources
- Challenges and examples of writing type providers, and research challenges for future developments in the combination of languages and rich structured information sources

Requirement: Internet connectivity for demos (demos will be extracted from the tutorial at http://research.microsoft.com/apps/video/dl.aspx?id=150061)

Bios:

Don Syme is a Principal Researcher at Microsoft Research, Cambridge. He has been a key contributor to the design of the very widely used C# programming language, in C# 2.0 Generics, and is the architect, researcher and designer of the F# programming language, a strongly typed functional programming language and sponsored jointly by Microsoft Research and the Microsoft Developer Division. His research interests are about bringing data and services into the programming experience. http://research.microsoft.com/en-us/people/dsyme/

Evelyne Viegas is the Director of Semantic Computing at Microsoft Research, Redmond, U.S.A. In her current role Evelyne is building initiatives which focus on information seen as an enabler of innovation, working in partnership with universities and government agencies worldwide. In particular she is creating programs around computational intelligence research to drive open innovation and agile experimentation via cloud-based services; and projects to advance the state-of-the-art in knowledge representation and reasoning under uncertainty at web scale. http://research.microsoft.com/en-us/people/evelynev/

F# 3.0

Integrating functional programming with rich structured information sources

Don Syme, Principal Researcher, Microsoft Research, Cambridge, UK

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

With F# we want to help fix this…

The mechanism we're adding to F# is called Type Providers

Type Providers

=

Language Integrated Data and Services

But first…

What is F# and why should I care?

F# is...

...a **practical, functional-first programming language** that allows you to write **simple code** to solve **complex problems**.

F# and Open Source

F# 2.0 compiler+library open source drop

Apache 2.0 license

www.tryfsharp.org

http://blogs.msdn.com/dsyme

Example #1 (Power Company)

I have written an application to balance the national power generation schedule … for an energy company.

…the calculation engine was written in F#.

The use of F# to address the complexity at the heart of this application clearly demonstrates a sweet spot for the language … algorithmic analysis of large data sets.

Simon Cousins (Eon Powergen)

A Challenge

Task #1: The names of the Amino Acids, as data

Task #2: A Chemical Elements Class Library

Task #3: Repeat for all Sciences, Sports, Businesses, …

Language Integrated Web Data

demo

```
// 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/a
"{\"query\":"+query+"}"

    let request : HttpWebRequest = downcast WebReques
    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 would we do this today?

Note: F# still contains no data

Open architecture

You can write your own type provider

Language Integrated Data Market Directory

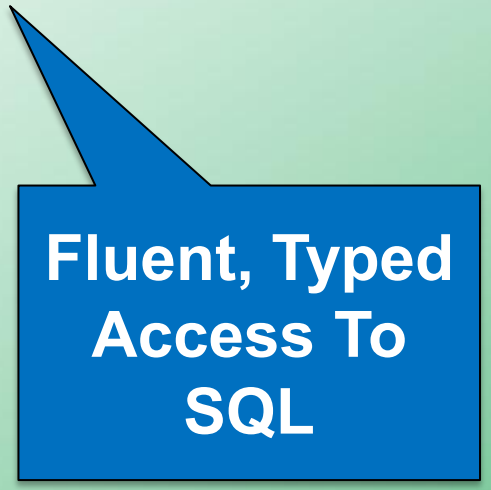
demo

F# + Semantic Web?

demo

SQL

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



SharePoint

```
type EmeaSite = SharePointSite<"http://myemea/">
```

Type Providers: Applications

- …web data
- …data markets
- …network management
- …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 programming needs to be information-rich too

The Type Provider Manifesto?

Consume anything! Directly!
Strongly typed! No walls!