Oliver Sturm

Information Rich Programming with F# 3.0

Oliver Sturm (@olivers)

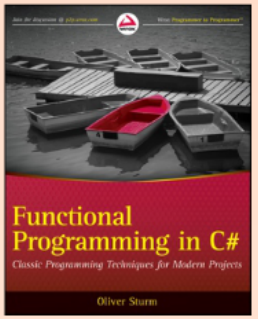
- Consultant, Trainer, Author
- Associate Consultant at thinktecture
- Course Author at Pluralsight

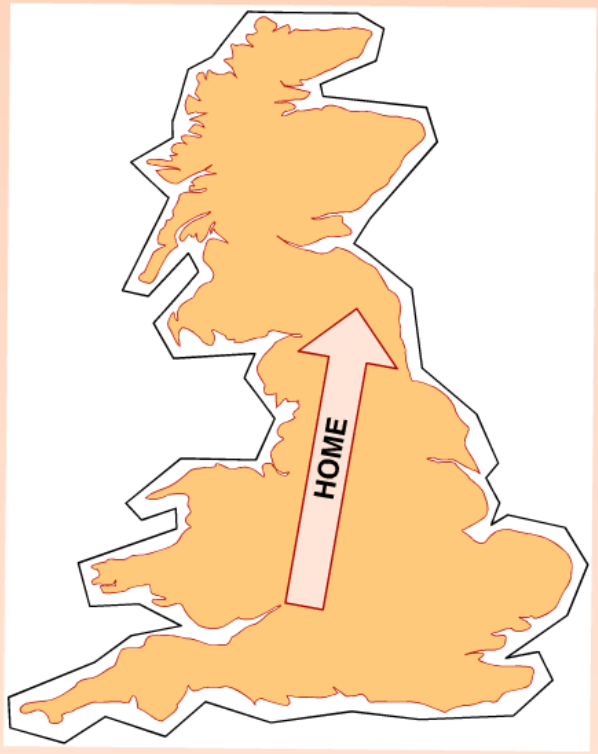
thinktecture

pluralsight
see what you can learn

DevExpress™ Expert

Microsoft Most Valuable Professional





Contact

oliver@oliversturm.com

Services

http://www.oliversturm.com

Agenda

- The new feature in F# 3.0: Type Providers
- Are we talking about dynamic programming?
- Looking at the standard providers

- Extensibility: (almost) the simplest custom type provider
- Checking out some custom providers in detail

What's the idea with Information Rich Programming?

Some points MS people like to make:

- Our world is information rich
- ... [skipping a few items here] ...
- Our programming languages are information sparse

In a nutshell:

Programming languages should make it easy
to interact with ever-changing data.


DEMO

F# Type Providers

- Plugins that supply information about types
- Type schema is loaded externally by the plugins
- Plugins interact with the F# compiler as well as Visual Studio (Intellisense)

Are we talking about dynamic programming?

- No.
- Not really.
- Type providers assume two things:
  - It is possible to define a schema for the data we're working with
  - The schema is widely fixed at compile time, i.e. it doesn't change at runtime based on application logic
- Type providers are more like automated code generation tools than dynamic programming in the usual sense
- But: many common and widely appreciated techniques used in dynamic languages and APIs comply with the assumptions above

Standard providers supplied with F# 3.0

- SqlDataConnection - access database through LINQ to SQL (SQLMetal)
  - DbmlFile - similar to SqlDataConnection, but using a previously created .dbml file
- SqlEntityConnection - access database through Entity Framework
  - EdmxFile - similar to SqlEntityConnection, but using a previously created .edmx file
- WsdlService - access web services that supply WSDL schema information
- ODataService - access web service through the OData protocol

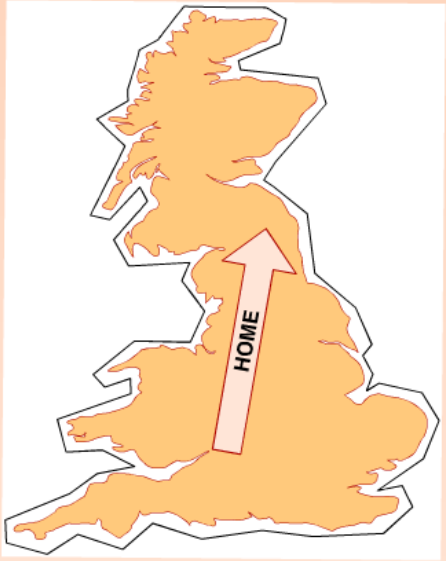

DEMO

Extensibility

- Type providers can be implemented through the two interfaces ITypeProvider and IProvidedNamespace
- Use attributes [<TypeProvider>] and [<TypeProviderAssembly>]
- Implementing the provided types themselves is not trivial - derive from System.Type
- MS are creating some useful code, but it's not complete and somewhat restricted


DEMO

Thank you for watching!



Contact

oliver@oliversturm.com

Services

http://www.oliversturm.com