

ADO.NET Entity Framework v4

Sven Aelterman

Lecturer & Web/Technology Specialist

Sorrell College of Business

Thank you, Jim, for

- Explaining what a Context is
- Showing how to use LINQ
- Admitting that LINQ to SQL is futureless

Presentation Overview

- What's new and different since v1?
- Getting Started: What You Need
- Demo 1: Getting Started
- Demo 2: Model First
- Demo 3: N-Tier
- Demo 4: POCO
- Wrap-Up & Q&A

Disclaimers

1. I don't work for Microsoft
I don't know what they'll do next.
2. This talk is uses pre-release products (B2)
What will it look like in the final version?
3. I don't do magic
Typos, Murphy, etc. cause demos to fail.
4. Demo code is not fit for any purpose other than entertainment
If you want production code, I charge \$125/hr to consult/mentor.

Changes Since v1

- Lazy Loading Support
- Model-First with DB Generation
- N-Tier
- POCO
- Text Template Transformation Toolkit
- Better stored procedure support

Getting Started

- .NET Framework 4 (currently Beta 2)
- Visual Studio 2010 (currently Beta 2)
- [Entity Framework Feature CTP 2](#)
 - Code Only enhancements (not demo'd)
 - Self-tracking entities template (Demo 3)
- T4 Editor (optional: syntax highlighting)
 - E.g. tangible engineering
(VS 2010 Extension Manager > Online)

Getting started with ADO.NET EF v4

- Creating a model based on an existing database
- Modifying the mappings in the model
- Work with non-entity stored procedure output

DEMO 1

Demo 1 Summary

- Skills from v1 transfer easily
- EF continues to support powerful mapping scenarios
- New/updated
 - Generating model from database now includes foreign key properties
 - Lazy loading enabled by default
 - Map stored procedure result set to complex type and method
- Disconnected scenario still requires DB roundtrip or setting properties to modified

Using a model-first approach (DDD)

DEMO 2

N-Tier and ADO.NET EF

- V1: No good solution (object context tracks changes)
 - Roundtrip to database and
 - Mark all properties as modified (using lots of custom code)
- V4: Support for
 - Self-Tracking Entities
 - Objects can track their own state changes
 - Requires control of client and server
 - POCO (Demo 4)
 - Requires more hand coding
 - Also consider ADO.NET Data Services



What is POCO?

Using ADO.NET EF v4 in a disconnected scenario using Self-Tracking Entities (STE)

- Designed for WCF
- Also works N-Tier, etc.

DEMO 3

Demo 3 Summary

- Using T4 Templates, EDM can be translated into “any” code
 - T4 Templates can be [customized](#) (not demo'd)
 - Obtain STE T4 Template from ADO.NET EF4 Feature CTP 2
 - T4 Templates can be shared (community, anyone?)
- Lazy loading affects n-tier development
- What if you don't control the client?

POCO Entities

- Create an Entity Data Model w/out code generation
- Write your own entity classes (or use T4)
- Write your ownObjectContext

DEMO 4

Demo 4 Summary

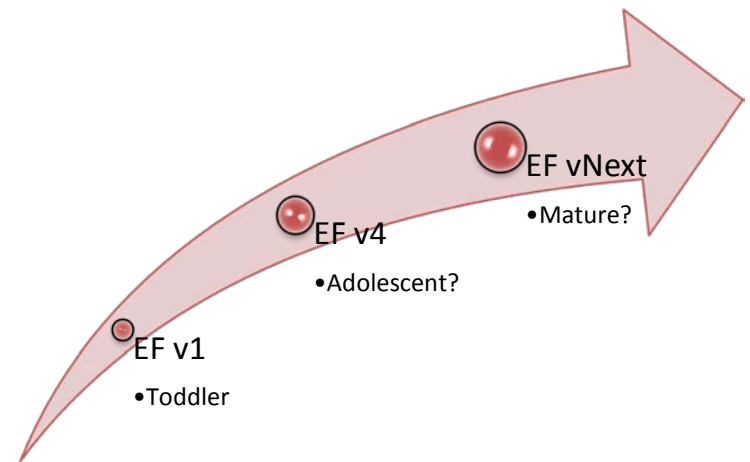
- POCO still requires a model
- Entities have to match the model (convention over code)
- Entities do not have to be hand-coded (T4)
- [Specific requirements](#) exist if you want to use deferred loading and change tracking

Session Summary

- Entity Framework v4 new features
 - Code generation supports self-tracking entities and POCO (*DIY at this time*)
 - Improved N-Tier support
 - Improved models and model development
- Conclusion from my v1 talk:
“Do not use in production”
- Conclusion from my v4 talk:
“Consider for small-scale projects”
- POCO enables TDD

Conclusions

- EF continues to be major part of MS data access strategy
 - Oslo
 - ADO.NET Data Services
- Progress of EF has been significant
- But Microsoft still needs us to provide feedback



Sven Aelterman

sven@adduxis.com

<http://www.adduxis.com> for full code, slides, and scripts

Q&A