Entity Framework (V4.0)

박용준(redplus@redplus.net)

MCT(Microsoft Certified Trainer)
Microsoft MVP(Most Valuable Professional)
닷넷코리아(http://www.dotnetkorea.com/) 운영자

엔터티 프레임워크 시작하기

엔터티 프레임워크 4.0 선수학습

- ASP.NET
- ADO.NET
- LINQ

데이터 접근 기술 개요

지난 10년의 MS Data Access 기술

- 뭐뭐뭐~~~
  - ADO.NET
  - SqlHelper
  - Enterprise Library
  - LINQ to SQL
  - WCF RIA Services
  - ADO.NET Entity Framework
- 결론은?
  - ADO.NET Entity Framework

ADO.NET History

ADO.NET < .NET 3.5

- Database Commands & Connections
- Data Readers
- (Typed) Data Sets
- Data Adapters / Table Adapters

ADO.NET in .NET 3.5

- Language Integrated Query
- Strongly typed queries and results
- LINQtoSQL and other LINQ implementations
- Entity Framework didn't make it!

ADO.NET in .NET 3.5 SP1

- **Entity Framework v. 1.0**
- Lazy Loading disabled by default
- Fat Objects
- Many imperfections
- LINQtoSQL remained the standard

ADO.NET in .NET 4.0

Entity Framework v. 4.0

LINQ(Language INtegrated Query)

- LINQ to SQL
  - SQL Server DB 처리
- LINQ to Entity
  - EDM 연동
- LINQ to XML
  - 인 메모리 XML 데이터 처리
- LINQ to Object
  - 컬렉션 개체 데이터 처리
  - IEnumerable, IEnumerable<T>

Entity Framework vs LinqToSQL

LinqToSQL

- Specific to Microsoft SQL Server
- Maps 1-to-1 to database tables, views, stored procedures and functions
- Ideal for quick data access construction to relatively well designed SQL Server databases
- Supports Table per Hierarchy Inheritance

Entity Framework

- Storage Provider Independent
- Supports many types of Object Relational Mappings, including many-to-many relationships
- Ideal for building conceptual models that aggregate a variëty of tables, sources, service etc. into a mash-up domain model
- Supports also Table per Subclass, Entity Splitting, Horizontal Splitting, and more…

ADO.NET Entity Framework

- Entity Data Model
  - 물리적 모델(SQL)과 개념적 모델(.NET)을 매핑

ADO.NET Entity Framework 4.0

- Model-first development
- Automatic pluralization
- Foreign keys in models
- POCO class support
- Lazy loading
- T4 Code Generation
- Template customization
- IObjectSet
- Virtual SaveChanges
- ObjectStateManager control
- Self-tracking entities
- SQL generation improvements
- More LINQ operator support
- LINQ extensibility
- ExecuteStoreQuery
- ExecuteStoreCommand
- SPROC import improvements
- Model defined functions
- WPF designer integration
- Code-Only development

읽어보기

- Entity Framework Q&A
  - http://msdn.microsoft.com/ko-kr/magazine/cc507640.aspx

실습 : Entity Framework 프로젝트 만들기

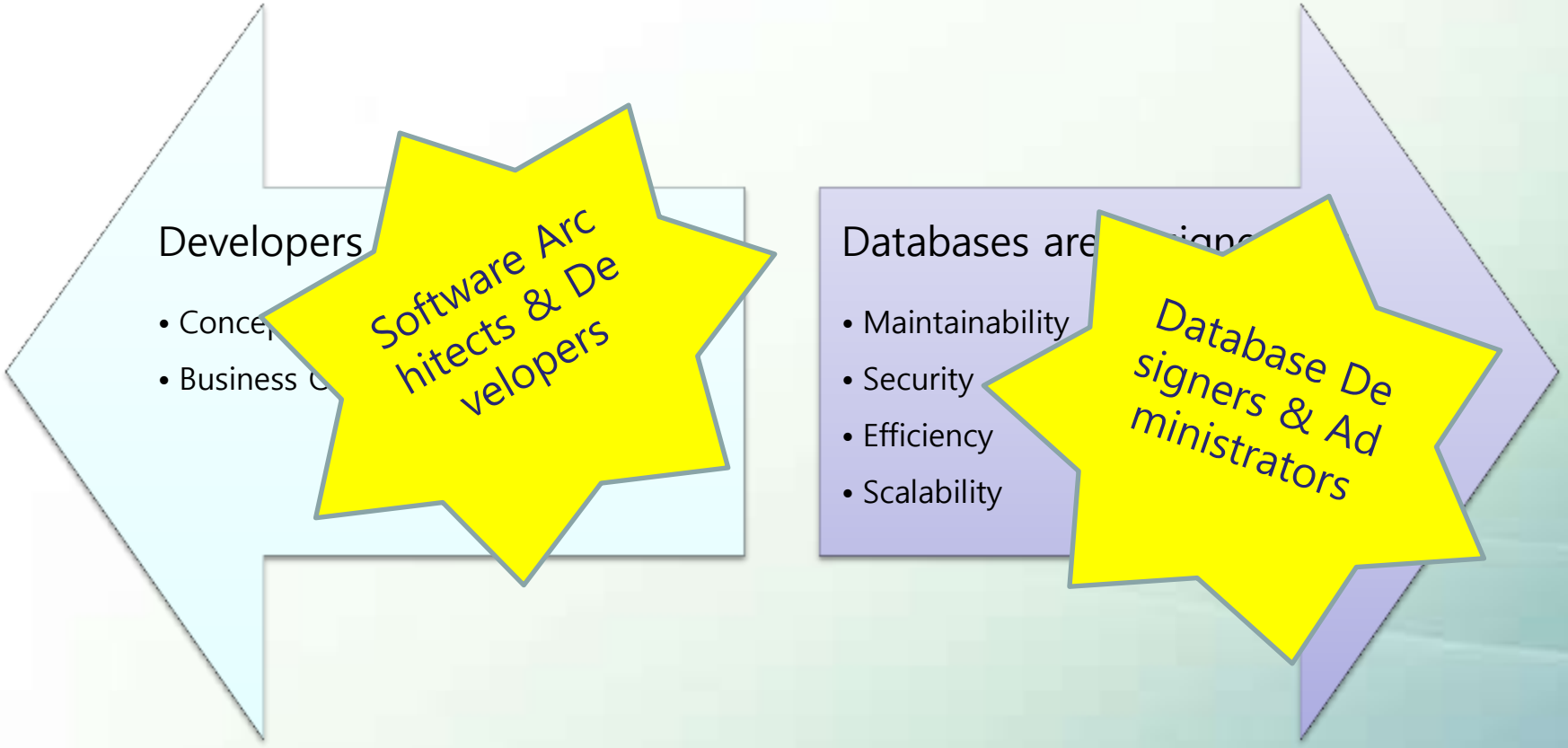
- C:₩EntityFramework₩WebEntityFramework
  - ASP.NET 4.0 웹 응용 프로그램

Entity Data Models

Data Models

- 데이터 모델 3가지
  - 물리적 모델(physical model)
    - 어디에 데이터가 저장되는지?
  - 논리적 모델(logical model)
    - 물리적 모델을 구현
    - 데이터가 테이블에 저장되는지?
  - 개념적 모델(conceptual model)
    - 논리적 모델과 매핑
- Entity Framework에서 개념적 모델을 구현해 놓은 것이 바로 Entity Data Model(EDM)이다.

딜레마(dilemma)


Developers
• Concep
• Business C
Software Architects & Developers
Databases are
• Maintainability
• Security
• Efficiency
• Scalability
Database Designers & Administrators

모델 디자인 정책 3가지

- Database-First Development(Design)
  - 미리 만들어진 DB를 기반으로 EDM 생성
- Model-First Development
  - Design conceptual model
  - Generate DDL to create
- Code-Only Development
  - Infer conceptual model and DDL directly from your class model
  - No .edmx file!

Entity Model Designer

- Visual Studio 2010에서 제공되는 Entity Designer
  - Mapping Details
  - Model Browser
- EDM Generator
  - EdmGen.exe : 명령어 기반의 디자인 도구

XML로 생성되는 모델

- .edmx 파일
  - XML로 구성됨 : Entity Designer에서는 그래픽 환경
  - SSDL 영역 : storage model
    - store schema definition language
  - CSDL 영역 : conceptual model
    - conceptual schema definition language
  - C-S 매핑 영역

모델에 엔터티 및 관계 추가

- Entity Designer에서 직접 엔터티 추가
  1. 엔터티 디자이너에서 마우스 오른쪽 버튼 클릭 후 엔터티 추가
  2. 스칼라 속성 추가
  3. 이름과 타입 지정
- Navigation 속성
  - 현재 엔터티와 관계가 있는 엔터티에 손쉽게 접근
- Model First Development 구현

실습 : 세미나 모델 : Rooms, Talks, Speakers

- ₩WebEntityFramework₩Models₩Seminar.edmx
  - Rooms : Talks와 일대다
    - Id
    - Name
  - Talks : Speakers와 다대다
    - Id
    - Title
    - When
  - Speakers
    - Id
    - Name

모델로부터 T-SQL 구문 생성하기

- 완성된 모델로부터 T-SQL 생성
  - SQL Server 2005 이상
  - Database Schema Name 속성 지정
    - 기본값 : dbo

실습 : 모델에서 T-SQL 뽑아내기

- ₩WebEntityFramework₩Models₩Seminar.sql

Database First	• “database is the truth” • why? it already exists, or you want low level control over the database • what? import model into edmx and tweak
Model First	• “edmx is the truth” • why? you want separation from code and database in a declarative format • what? create a model and tweak
Code First	• “code is the truth” • why? primarily focused on code shape, database is an implementation detail • what? define classes in code, adjust shape using contextbuilder

모델에 저장 프로시저 추가

- EDM에 손쉽게 SP 추가 가능
- SP는 EDM에서 메서드로 표현되고, SSDL에 정의
- SP는 Entity Designer에서는 보여지지 않고 Model Browser 영역에서 보여짐

저장 프로시저(Stored Procedures) 사용

Stored Procedure and Mapping in EDM

```
CREATE PROCEDURE [Sales].[InsertRewardsClaim]
  @claimID [int],
  @pointsUsed [int],
  @rewardID [int],
  @contactID [int] AS
BEGIN
  SET NOCOUNT ON;
  INSERT INTO [Sales].[RewardsClaimed]
  ([ClaimID], [PointsUsed], [RewardID], [ContactID])
  VALUES (@claimID, @pointsUsed, @rewardID, @contactID)
END;
```

```
<ModificationFunctionMapping>
    <InsertFunction FunctionName=
        "AdventureWorksModel.Store.InsertRewardsClaim" >
      <ScalarProperty Name="ContactID" ParameterName="contactID" />
      <ScalarProperty Name="RewardID" ParameterName="rewardID" />
      <ScalarProperty Name="PointsUsed" ParameterName="pointsUsed" />
      <ScalarProperty Name="ClaimID" ParameterName="claimID" />
    </InsertFunction>
</ModificationFunctionMapping>
```

실습 : 저장 프로시저 호출하기

- ₩WebEntityFramework
  - ₩Views₩UsingStoredProcedure.aspx

새 개체 생성 후 저장 하기 : AddObject()

```
Contact contact = new Contact();

Contact.LastName = "RedPlus";
...

entities.AddToContacts(contact);
```

첫번째 방법 :
개체 생성 후 Add 메서드 호출

```
Contact contact = Contact.CreateContact(0, true,
    "Ronald", "Adina", 0, "xyz", "abc",
    Guid.NewGuid(), DateTime.Now, 1000);

entities.AddToContacts(contact);
```

두번째 방법 :
Create() 메서드 호출

```
entities.SaveChanges();
```

Save the changes to the database

Hawaso

실습 : 입력(저장) 패턴

- ₩WebEntityFramework
  - ₩Views₩FrmTaskListsInsert.aspx

엔터티 업데이트(수정)

```
EntityKey key = entities.CreateEntityKey("StoreContacts", contact);

object storeContactToModify;

if (entities.TryGetObjectByKey(key, out
        storeContactToModify))
{
        entities.ApplyCurrentValues(key.EntitySetName,
        contact);
}
```

Get the EntityKey of the entity to update

Load the entity to update into the context

Copy changed property values from a detached entity object

```
entities.SaveChanges();
```

Save the changes to the database

Hawaso

실습 : 수정 패턴

- ₩WebEntityFramework
  - ₩Views₩FrmTaskListsUpdate.aspx

엔터티 삭제

```
object storeContactToDelete = null;

EntityKey key = new
 EntityKey("AdventureWorksEntities.StoreContacts",
    "contactID", contactID);

if (entities.TryGetObjectByKey(key, out
 storeContactToDelete))
{
        entities.DeleteObject(storeContactToDelete );
}
```

Create the EntityKey of the entity to update

Load the entity to delete into the context

Delete the entity

```
entities.SaveChanges();
```

Save the changes to the database

Hawaso

실습 : 삭제 패턴

- ₩WebEntityFramework
  - ₩Views₩FrmTaskListsDelete.aspx

TODO : 더 많은 Entity Framework 4 학습

- 웹 캐스트
  - http://www.Entity Framework.net/learn/
  - http://channel9.msdn.com/

감사합니다

데브렉(http://www.devlec.com/) : 쉽게 배우는 Entity Framework 4

감사합니다.
박용준