Emerging Data Technologies

Dive into Entity Framework 4.0

Mike Benkovich
Microsoft Corporation
Mike.Benkovich@Microsoft.com
http://blogs.msdn.com/benko
Twitter: @mbenko
http://www.BenkoTIPS.com

Microsoft® .NET Framework

Where can I get more info?

- Visit my site – www.BenkoTIPS.com
  - Resources from today's talk
  - Webcasts
  - Downloads
  - More!
- Subscribe to my blog (my boss will love that ☺)
  - http://blogs.msdn.com/benko
- Register for MSDN Events at www.msdnEvents.com

- Have an office full of developers who couldn't make it?

  → Ask me about **MSDN OnSite Events**

Take aways

- Understand the motivation of Entity Framework 4.0
- Explore mapping and runtime features and services
- See the technology in action…DEMOS!

Entity Framework 4.0

- Provides tools to support Entity Data Model (EDM)
- Tools and services to support building clients
- Provide a means to go from objects to physical

Why an ORM?

```
// Create a connection with the AdventureWorks connection string.
using (SqlConnection conn = new SqlConnection(
    ConfigurationManager.ConnectionStrings["AdventureWorks"].ConnectionString))
{
    conn.Open();

    // Create a command to join Customer and CustomerContactInfo tables.
    SqlCommand command = conn.CreateCommand();
    command.CommandText = @"
        SELECT  cust.FirstName, cust.LastName, contact.EmailAddress
        FROM    [dbo].[Customer] AS cust
        JOIN    [dbo].[CustomerContactInfo] AS contact
        ON      cust.CustomerID = contact.CustomerID
        WHERE   cust.MiddleName IS NULL";

    // Execute the command and obtain a data reader.
    using (SqlDataReader reader = command.ExecuteReader(
        CommandBehavior.SequentialAccess))
    {
        while (reader.Read())
        {
            // Write a response line using values from the reader.
            Response.Write(String.Format("<p>{0}\t{1}\t{2}</p>",
                reader["FirstName"],
                reader["LastName"],
                reader["EmailAddress"]));
        }
    }
}
```

Source: Eric Nelson – DEV305 - PDC09

Why an ORM?

```
using (AdventureWorksModel model = new AdventureWorksModel())
{
    var query = from c in model.Customer
                where c.MiddleName == null
                select new {
                    FirstName = c.FirstName,
                    LastName = c.LastName,
                    EmailAddress = c.EmailAddress };

    foreach (var c in query)
    {
        Response.Write(String.Format("<p>{0}\t{1}\t{2}</p>",
            c.FirstName,
            c.LastName,
            c.EmailAddress));
    }
}
```

Source: Eric Nelson – DEV305 - PDC09

Entity Framework 1.0 Features

- Generate model from database schema
  - Very rich mapping layer
  - Inheritance, rename, aggregate, filter
- Simplified CRUD operations
- LINQ querying*
- Worked with many databases
- Database agnostic query language
- SP support for data retrieval/modification*
- Designer*
- Strategic!

Entity Framework 1.0 Pain Points…

- The designer
- Model First not supported
- Poor support for Stored Procedures
- No Pluralization/Singularization
- Foreign keys hidden
- Lazy loading not supported
- Missing LINQ Operators vs LINQ to SQL
- Generated SQL unreadable
- No support for POCO
- N-Tier difficult…

Entity Framework 4.0

- Better tools & design experience
- More powerful and flexible runtime
- AND
  - N-Tier
  - Persistence Ignorance
  - Code only support
- Built into .NET Framework 4.0

Better Tools and Design Experience

- Model First
- Templated code generation
- Stored Procedures
- Pluralization/Singularization
- Complex Types
- Better delete & search

Hello Entity Framework 4.0

Model First
Deploy to SQL Azure

Dive into Mapping

- **Inheritance**
  - **Table per Hierarchy**
  - Table per Type
  - Table per Concrete Type
  - Hybrids
- **Many entities to one table**
- **Stored Procedures**
- Many tables to one entity
- Abstract Entities
- Associations within EntitySets
- Associations across EntitySets
- Store-side discriminators
- EDM-side discriminators
- QueryViews, Defining Query, CommandText

DEMO

Mapping features

Runtime Flexibility & Power

- Deferred Loading (lazy loading)
- Foreign Keys
- More complete implementation of LINQ
- ExecuteStoreQuery
- EntityFunctions and SqlFunctions
- Improvements to generated SQL

Runtime

Deferred (Lazy) loading

Why POCO

- Testability
- Agility
- Freedom from the tyranny of oppressive frameworks
- Serialization concerns

Defining POCOs

- The POCO class name must match name of the entity type
- For every entity type property
  - There must be a corresponding prop in the POCO
  - The data types must match
  - The name must match

POCO Proxy

- Roles
  - Change Tracking
  - Relationship fix-up
  - Lazy Loading
- Serialization
  - Disable or
  - ProxyDataContractResolver

N-Tier Improvements

- EF4 N-Tier improvements:
  - New low level state management API
  - Self-Tracking Entities
- Best practices:
  - Use standard change tracking when possible
  - Use Self-Tracking Entities if sharing entity types is acceptable
  - Create data transfer objects when low coupling is needed
- Other technologies to learn about:
  - WCF Data Services - bit.ly/learndataservices
  - WCF RIA Services - bit.ly/learnriaservices

N-Tier (Self Tracking Entities)

Client

```
static bool ValidSTEUpdate(INorthwindService svc)
{
    var customer = svc.GetCustomer("ALFKI");

    // modify contact name of customer
    customer.ContactName += "+";

    // add a new order
    var newOrder = new Order();
    newOrder.Order_Details.Add(new Order_Detail()
    {
        ProductID = products.Where(p => p.ProductName
                    == "Chai").Single().ProductID,
            Quantity = 1
        });
    customer.Orders.Add(newOrder);

    return svc.SubmitOrder(newOrder);
}
```

Service

```
public class NorthwindService : INorthwindService
{
    public Customer GetCustomer(string id)
    {
        using (var ctx = new NorthwindEntities())
        {
            return ctx.Customers.Include("Orders")
                .Where(c => c.CustomerID == id)
                .SingleOrDefault();
        }
    }

    public bool SubmitOrder(Order newOrder)
    {
        using (var ctx = new NorthwindEntities())
        {
            ctx.Orders.ApplyChanges(newOrder);
            ValidateOrderGraph(ctx, newOrder);
            return ctx.SaveChanges() > 0;
        }
    }
}
```

Danny Simmons MSDN Article: Building N-Tier Apps with EF4

http://code.msdn.microsoft.com/mag200911EF4

http://msdn.microsoft.com/en-gb/magazine/ee335715.aspx

N-Tier & Self Tracking Entities

Summary

- Entity Framework 4.0 answer the pain points of 1.0
- New features enable broad scenarios for leveraging the technology

Additional Resources

- Eric Nelson Blog
  http://iupdateable.com
- MSDN Data Developer Center:
  http://msdn.com/data
- ADO.NET Team Blog:
  http://blogs.msdn.com/adonet
- OData Blog:
  http://odata.org/blog
- WCF Data Services Team Blog:
  http://blogs.msdn.com/astoriateam
- EF Design Blog:
  http://blogs.msdn.com/efdesign
- Patterns and Testability:
  http://bit.ly/learnef4test, http://bit.ly/ef4wpfsample