VSLive! NEW YORK

Architect LINQ to SQL End-to-end

Benjamin Day
Benjamin Day Consulting, Inc

Level: Advanced

MVP Microsoft Most Valuable Professional

About the speaker

BDC Benjamin Day Consulting

- Owner, Benjamin Day Consulting, Inc.
  - Email: benday@benday.com
  - Web: http://www.benday.com
  - Blog: http://blog.benday.com
- Trainer
  - Visual Studio Team System, Team Foundation Server
- Microsoft MVP for C#
- Microsoft VSTS/TFS Customer Advisory Council
- Microsoft Cloud Services Advisory Group
- Leader of Beantown.NET INETA User Group

2

VSLive! NEW YORK

© 2008 Benjamin Day Consulting, Inc. All rights reserved. This presentation is for informational purposes only. Benjamin Day Consulting, Inc. makes no warranties, express or implied, in this summary.

Agenda

- Build an n-tier application
  - Testable
  - Maintainable
  - Uses LINQ to SQL as the data access technology

3

VSLive! NEW YORK

Agenda: Questions to Answer

- How do I use LINQ to SQL in an n-tier application?
- How should my tiers be organized?
- What are the best-practices for business objects?
- How do I use a base class for my entities?
- How do I do change auditing?
- What are the best practices for unit testing?
- What are my options for concurrency management?
- How does LINQ to SQL affect my ASP.NET code?
  - Or, what do I have to worry about from ASP.NET?

4

VSLive! NEW YORK

© 2008 Benjamin Day Consulting, Inc. All rights reserved. This presentation is for informational purposes only. Benjamin Day Consulting, Inc. makes no warranties, express or implied, in this summary.

How do I use LINQ to SQL in an n-tier Application?

- Use it as an object-relational mapping (ORM) technology
  - Save, Update, Load, and Delete database records without much code
- Always abstract away the existence of your ORM technology as much as possible
  - Little -- ideally no -- visible LINQ to SQL code
  - This means you should hide your L2Q entities
- Interface-driven programming
  - More maintainable
  - Easier to test

5

VSLive! NEW YORK

How should my tiers be organized?

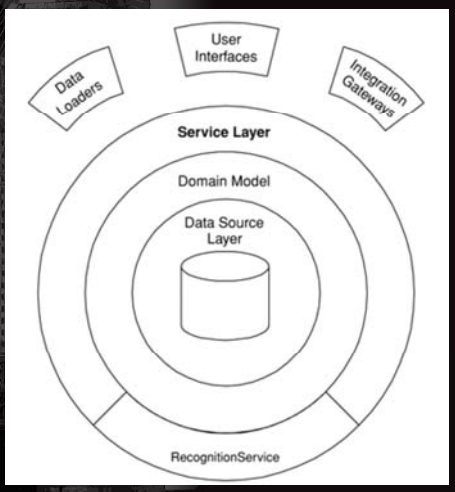
- My $0.02...
  - Use the Service Layer Pattern for the business tier
  - Model View Presenter (MVP) Pattern for the User Interface

6

VSLive! NEW YORK

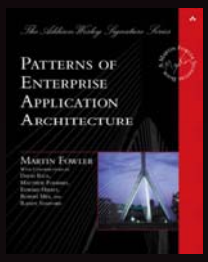
© 2008 Benjamin Day Consulting, Inc. All rights reserved. This presentation is for informational purposes only. Benjamin Day Consulting, Inc. makes no warranties, express or implied, in this summary.

Service Layer Pattern



"Defines an application's boundary with a layer of services that establishes a set of available operations and coordinates the application's response in each operation."

-Randy Stafford



From "Patterns Of Enterprise Application Architecture" by Martin Fowler, Randy Stafford, et al. Chapter 9

7

VSLive! NEW YORK

The Tiers

- Presentation tier
  - ASP.NET
  - Windows Forms
  - WPF
  - WCF Service
  - The "View" of MVP
- Presenter Tier
  - Handles the "conversation" between the presentation tier implementation and the business tier
  - Defines the "View" Interfaces
  - "Presenter" in MVP
- Business tier
  - Business object interfaces
  - Business objects
    - **LINQ to SQL entities**
    - **The "Model" in MVP**
  - Business facades
    - **Manipulate business objects**
    - **Handle requests for CRUD operations**
- Data Access Tier
  - LINQ to SQL as ORM
- Data Storage Tier
  - SQL Server

VSLive! NEW YORK

© 2008 Benjamin Day Consulting, Inc. All rights reserved. This presentation is for informational purposes only. Benjamin Day Consulting, Inc. makes no warranties, express or implied, in this summary.

Best Practices for Business Objects

- Use a common base class
  - Push common functionality into the base class
- Use a common base interface
- Each business object gets its own business interface

9

VSLive! NEW YORK

Best Practices for Business Objects

- No Load(), Save(), or Delete() methods on the business objects
  - The business objects are lightweight
  - POJOs - plain old Java objects
  - They're smart (well, not completely dumb) data transfer objects
  - CRUD methods/operations are taken care of by the business facades
- Business facades use the Service Layer/Transaction Script pattern
  - The facades define all the operations available through the business tier
  - Should be at least 1 unit test per business façade operation

10

VSLive! NEW YORK

© 2008 Benjamin Day Consulting, Inc. All rights reserved. This presentation is for informational purposes only. Benjamin Day Consulting, Inc. makes no warranties, express or implied, in this summary.

Best Practices for Unit Testing

- Test success and failure cases
- Provide descriptive error message on asserts
- Create a base class that all your unit tests inherit from
- Make tests autonomous
  - Avoid dependencies between tests
  - Tests should not have to run in any particular order
- Don't assume what data is in or not in the database
  - Not knowing may cause false passes or fails

11

VSLive! NEW YORK

Unit Testing: How do you KNOW what data is in the database?

- Make sure the database is empty before each test runs
- Any required data for the test is put into the database as part of the test

12

VSLive! NEW YORK

© 2008 Benjamin Day Consulting, Inc. All rights reserved. This presentation is for informational purposes only. Benjamin Day Consulting, Inc. makes no warranties, express or implied, in this summary.

Clearing the database: Option #1

- DELETE every row in each table before the test
  - Problem: You have to worry about FK constraints
  - Problem: you have to write/maintain the DELETE script

13

VSLive! NEW YORK

Clearing the database: Option #2

- Keep track of any row you write to the database and delete it after the test
- Bug prone
- Your business tier code could create records that you don't know about
- $0.02: you don't want or need to worry about all those rows
- Tough to analyze faults: If your test fails and you want to look at the data in the database after the test, it's not there.

14

VSLive! NEW YORK

© 2008 Benjamin Day Consulting, Inc. All rights reserved. This presentation is for informational purposes only. Benjamin Day Consulting, Inc. makes no warranties, express or implied, in this summary.

Clearing the database: Option #3

- Create a database TRANSACTION at the start of the unit test and call ROLLBACK at the end of the test
- Inserted data isn't written to disk until COMMIT is called
  - Can't be read from a different database connection because the records are locked
  - Can't verify the data using a separate connection
- Tough to analyze faults: If your test fails and you want to look at the data in the database after the test, it's not there.

15

VSLive! NEW YORK

There's got to be a better way.

- LINQ to SQL to the rescue!

16

VSLive! NEW YORK

© 2008 Benjamin Day Consulting, Inc. All rights reserved. This presentation is for informational purposes only. Benjamin Day Consulting, Inc. makes no warranties, express or implied, in this summary.

DataContext Saves The Day

- DataContext can manage the schema for you
  - Enables Model-First development
- DataContext methods
  - DatabaseExists()
  - DeleteDatabase()
  - CreateDatabase()

17

VSLive! NEW YORK

Unit Tests and DataContext Schema Management

- In your unit test base class
  - Define TestInitialize() method and mark it with [TestInitialize()] attribute
  - In TestInitialize(), call DeleteDatabase() then CreateDatabase()
- Any records from previous test executions are gone
- Data is still there after the test executes
- Side benefit: your database is always in sync with the business object design
  - No hunting down object-relational schema mismatch errors

18

VSLive! NEW YORK

© 2008 Benjamin Day Consulting, Inc. All rights reserved. This presentation is for informational purposes only. Benjamin Day Consulting, Inc. makes no warranties, express or implied, in this summary.

Code Demo 1

- Create a unit test project
- Create a business project
  - Add Entities
- Create a unit test base class
- Add TestInitialize() to base class
- Write a unit test to create a Person

19

VSLive! NEW YORK

The Business Tier

VSLive! NEW YORK

© 2008 Benjamin Day Consulting, Inc. All rights reserved. This presentation is for informational purposes only. Benjamin Day Consulting, Inc. makes no warranties, express or implied, in this summary.

Best Practice: Hide the Implementation Details

- Right now, LINQ to SQL is fully exposed
- We're exposing the entities
- We're relying on non-business tier code to do the data access calls

21

VSLive! NEW YORK

Best Practice: Hide the Implementation Details

- Business tier should be more service-based
  - Create Person
  - Save Person
  - Get List of Person objects
  - Get Person By Id
- Why?
  - We can test that functionality
  - We can re-use the functionality
  - Calling code doesn't have to worry about the implementation details --> it just works
- How?
  - Create a business façade object

22

VSLive! NEW YORK

© 2008 Benjamin Day Consulting, Inc. All rights reserved. This presentation is for informational purposes only. Benjamin Day Consulting, Inc. makes no warranties, express or implied, in this summary.

Code Demo 2

- Create a PersonFacade that wraps the data context operations
- Create unit tests and implementations for
  - Create Person
  - Save Person
  - Get List of Persons
  - Get Person By Id

23

VSLive! NEW YORK

Best Practice: Hide the Implementation Details

- We're still exposing the LINQ to SQL entities
- Makes it difficult for us to modify the entities
  - Changes to the entities ripple out of the business tier
- We should hide the entities behind interfaces

24

VSLive! NEW YORK

© 2008 Benjamin Day Consulting, Inc. All rights reserved. This presentation is for informational purposes only. Benjamin Day Consulting, Inc. makes no warranties, express or implied, in this summary.

Code Demo 3

- After the demo, LINQ to SQL will be wrapped inside of the Business Tier
- Create partial class implementation for Person
- Create IPerson interface
- Make Person implement IPerson
- Make PersonFacade use IPerson instead of Person
- Update unit tests to use interfaces

25

VSLive! NEW YORK

Best Practice: Common Code Should Be In a Base Class

- We've just been handed some requirements:
  - All business objects should record
    - **Who created them**
    - **When were they created**
    - **Who last modified them**
    - **When were they last modified**
  - All business objects should have a primary key property "Id"
  - All business objects should implement optimistic concurrency
    - **Easiest way is to do this using the LastModified column**

26

VSLive! NEW YORK

© 2008 Benjamin Day Consulting, Inc. All rights reserved. This presentation is for informational purposes only. Benjamin Day Consulting, Inc. makes no warranties, express or implied, in this summary.

Business Base with LINQ to SQL

- Ideal BusinessBase implementation

```
public abstract class BusinessBase
{
    public virtual int Id { get; set; }
    public virtual DateTime CreatedDate { get; set; }
    public virtual DateTime ModifiedDate { get; set; }
    public virtual string CreatedBy { get; set; }
    public virtual string ModifiedBy { get; set; }
}
```

- Property implementations are in BusinessBase
- Too bad that LINQ to SQL won't let you
- Can't "map" the properties

27

VSLive! NEW YORK

Business Base with LINQ to SQL

- Make the properties "abstract"

```
public abstract class BusinessBase
{
    /*
     * This won't work
     *
    public virtual int Id { get; set; }
    public virtual DateTime CreatedDate { get; set; }
    public virtual DateTime ModifiedDate { get; set; }
    public virtual string CreatedBy { get; set; }
    public virtual string ModifiedBy { get; set; }
    */

    public abstract int Id { get; set; }
    public abstract DateTime CreatedDate { get; set; }
    public abstract DateTime ModifiedDate { get; set; }
    public abstract string CreatedBy { get; set; }
    public abstract string ModifiedBy { get; set; }
}
```

- "Abstract" forces derived classes to implement those properties

28

VSLive! NEW YORK

© 2008 Benjamin Day Consulting, Inc. All rights reserved. This presentation is for informational purposes only. Benjamin Day Consulting, Inc. makes no warranties, express or implied, in this summary.

How do I populate the business base fields?

- The DataContext has partial methods for each entity
  - Insert*EntityName*()
  - Update*EntityName*()
  - Delete*EntityName*()
- Provide implementation for these methods
- Populate the values
- WARNING: Don't forget to call ExecuteDynamicInsert(), ExecuteDynamicUpdate(), and/or ExecuteDynamicDelete()
  - If you provide a partial implementation for above methods but don't call ExecuteDynamic*Xyz*() then NOTHING will get written to the database.

29

VSLive! NEW YORK

Code Demo 4: Common Base Class

- Create BusinessBase class
- Create IBusinessBase interface
- Create partial implementations to populate the values
- Adjust “Inheritance Modifier” on the entities
- Create tests to verify business base logic
- HINT: Notice that when we run the tests that the database schema gets updated

30

VSLive! NEW YORK

© 2008 Benjamin Day Consulting, Inc. All rights reserved. This presentation is for informational purposes only. Benjamin Day Consulting, Inc. makes no warranties, express or implied, in this summary.

The User Interface

VSLive! NEW YORK

User Interfaces: The Redheaded Stepchild of the Unit Testing World

- Not easy to automate the UI testing
- Basically, automating button clicks
- UI's almost have to be tested by a human
  - Computers don't understand the "visual stuff"
  - Colors, fonts, etc are hard to unit test for
  - "This doesn't look right" errors
- The rest is:
  - Exercising the application
  - Checking that fields have the right data
  - Checking field visibility

32

VSLive! NEW YORK

© 2008 Benjamin Day Consulting, Inc. All rights reserved. This presentation is for informational purposes only. Benjamin Day Consulting, Inc. makes no warranties, express or implied, in this summary.

Best Practice: UI Unit Testing

- Solve the problem by not solving the problem
- Minimize what you can't automate

33

VSLive! NEW YORK

User Interface Testing: The Presenters & Views

- Model View Presenter (MVP) models the conversation between UI implementation and the Business Tier
- For each real UI that you want, create a C# "view" interface
  - Any fields that appear on the screen go on this interface
- Presenter Façade will know how to
  - Load, Save business objects
  - Populate the UI from the business objects
  - Populate the business objects from the UI
- Business & UI Tiers abstracted from each other

34

VSLive! NEW YORK

© 2008 Benjamin Day Consulting, Inc. All rights reserved. This presentation is for informational purposes only. Benjamin Day Consulting, Inc. makes no warranties, express or implied, in this summary.

Unit Testing the MVP

- Create a "stub" implementation of the “view” interfaces
  - IPersonSearch
  - IPersonDetail
- The “Presenter” = Editor Façade
- Unit test the Editor Façade methods that manipulate the C# interfaces
  - InitializeBlank()
  - LoadById()
  - Save()
  - Search()

35

VSLive! NEW YORK

Best Practice for MVP “View” Design

- Avoid referencing business objects or business interfaces on “View” interfaces
- Favor scalar types
- If necessary, create UI specific interfaces
  - If the business tier has “IPerson”
  - Create “IPersonInfo” for UI
  - Keeps needs of the UI separate from the needs of the Business Tier
- ASP.NET: This almost guarantees your application is stateless

36

VSLive! NEW YORK

© 2008 Benjamin Day Consulting, Inc. All rights reserved. This presentation is for informational purposes only. Benjamin Day Consulting, Inc. makes no warranties, express or implied, in this summary.

Code Demo 5

- Implement PersonEditorFacade, IPersonSearch, IPersonDetail
- Test functionality via stubs
- Create an ASP.NET implementation

37

VSLive! NEW YORK

Why is this more testable?

- Each page/screen only has to get/set the value from interface property into the right display control
- UI does not know anything about business objects
- Doesn't know about any details of loading or saving
- Doesn't have to know about validation
- All this logic goes into the editor façade and testable via unit test

38

VSLive! NEW YORK

© 2008 Benjamin Day Consulting, Inc. All rights reserved. This presentation is for informational purposes only. Benjamin Day Consulting, Inc. makes no warranties, express or implied, in this summary.

Cleanup and ASP.NET

VSLive! NEW YORK

Clean Up After Yourself

- Best practice: If you're done with an object and it's IDisposable, call Dispose()
  - Or wrap the reference in a using() { }
- DataContext implements IDisposable
- DataContext needs to get rid of its database connection

40

VSLive! NEW YORK

© 2008 Benjamin Day Consulting, Inc. All rights reserved. This presentation is for informational purposes only. Benjamin Day Consulting, Inc. makes no warranties, express or implied, in this summary.

DataContext Cleanup is Tricky in ASP.NET

- You don't exactly know when you're done with the DataContext
- If you call Dispose() too soon,
  - Lazy loaded collections may fail or error
  - Unusual behavior and lost saves

41

VSLive! NEW YORK

Handle Cleanup In ASP.NET

- Share a single DataContext per HTTP Request
  - Sharing a single DataContext is helpful for non-ASPX apps, too
- Close it at the end of the request
- How do you know when the request is finished?
  - Create an HTTP Module (IHttpModule)
  - HTTP Module hooks in to ASP.NET and is notified when the request is finished
  - Call DataContext.Dispose() from the Close() method on the http handler

42

VSLive! NEW YORK

© 2008 Benjamin Day Consulting, Inc. All rights reserved. This presentation is for informational purposes only. Benjamin Day Consulting, Inc. makes no warranties, express or implied, in this summary.

Code Demo 6: HTTP Module

- Create an HTTP Module
- Implement Close()
- Hook it in to the ASP.NET application using Web.config

43

VSLive! NEW YORK

Summary

- N-Tier application
- Testable
  - Database is consistent for each test
  - Tested business tier
  - Tested user interface functionality
- LINQ to SQL is hidden
  - You could change out the ORM implementation with ease
  - No “leaky abstraction”
- Handling DataContext management in ASP.NET
  - Releasing the connection reliably

44

VSLive! NEW YORK

© 2008 Benjamin Day Consulting, Inc. All rights reserved. This presentation is for informational purposes only. Benjamin Day Consulting, Inc. makes no warranties, express or implied, in this summary.

Questions?

45

VSLive! NEW YORK

About the speaker

BDC Benjamin Day Consulting

- Owner, Benjamin Day Consulting, Inc.
  - Email: benday@benday.com
  - Web: http://www.benday.com
  - Blog: http://blog.benday.com
- Trainer
  - Visual Studio Team System, Team Foundation Server
- Microsoft MVP for C#
- Microsoft VSTS/TFS Customer Advisory Council
- Microsoft Cloud Services Advisory Group
- Leader of Beantown.NET INETA User Group

46

VSLive! NEW YORK

© 2008 Benjamin Day Consulting, Inc. All rights reserved. This presentation is for informational purposes only. Benjamin Day Consulting, Inc. makes no warranties, express or implied, in this summary.