



Taking Customer Service to the Next Level



The Joys and Pains of a Long Lived Codebase

Jeremy D. Miller

November 20th, 2008

About Me

- Chief Software Architect at Dovetail Software in Austin, TX
- Author of StructureMap
 - ◆ <http://structuremap.sourceforge.net>
- “The Shade Tree Developer”
 - ◆ <http://codebetter.com/blogs/jeremy.miller>
- jeremydmiller@yahoo.com

What is StructureMap?

- Inversion of Control / Dependency Injection tool for .Net
- Open Source under the Apache 2.0 license
- I'm the primary author

...this isn't really about StructureMap itself

The Journey So Far

- Recovering data centric VB6 developer to...
- StructureMap was originally released in 2004
- TDD was new in .Net circles
- .Net 1.1 to .Net 2.0 to .Net 3.5
- The “Python 3000” Release in 2008

What has Changed?

- Lower tolerance for ceremony
- Programmatic configuration is now favored over Xml configuration
- More complicated usage
 - ◆ Composite configuration
 - ◆ Interception
- Convention over Configuration
- Explicit Configuration vs. Just Let it Work
- Language innovation

What Have I Learned?

- Solid design in the small is crucial
- To organize and structure by behavior
- Automated test coverage isn't automatically effective
- Some nasty lessons about creating API's for other developers

Don't Repeat Yourself

- It's 2005, and Generics are all the rage in .Net
- Wouldn't it be nice if StructureMap could support open generic types...

- Lesson Learned: Even innocuous looking duplication can be harmful



Taking Customer Service to the Next Level



Encapsulation is Key

Vile, Evil Code

```
public class ClassWithFoos
{
    private readonly IDictionary<string, Foo> _foos
        = new Dictionary<string, Foo>();

    public IDictionary<string, Foo> Foos
    {
        get { return _foos; }
    }
}
```

Keep the Code Limber

- Watch for similar code
- Use static code analysis as another pair of eyes
- Reduce the size of the codebase

Better Abstractions

- “Noun” based design is naïve
- Design abstractions by responsibilities and roles
- Use object role stereotypes

“The Great Refactoring of Aught Eight”

Before

- InstanceManager
- InstanceFactory
- InstanceMemento

After

- Container
- InstanceFactory (simpler)
- Instance
- BuildSession
- PipelineGraph
- BuildPolicy



Taking Customer Service to the Next Level



Test Driven Development allows for
modifiability...

...if done well

Organizing Tests

- 1 Concrete Class == 1 TestFixture (Not!)
- TestFixture per Feature
- Arriving at Behavior Driven Development
 - ◆ Number of Assertions per Test
 - ◆ Pay attention to the “language” of tests
 - ◆ Emphasize scenarios over units of code
 - ◆ Test what you’re testing

Test Input

- Ideally, tests should be self-contained
- Use meaningful test data
- Watch the Expensive Test Setup Smell
- Invest in ObjectMothers or Test Data Builders
- Be extremely cautious in reusing test data across multiple tests

Building Configuration Intensive Frameworks

- Isolate functionality from configuration
- Isolate the user from framework internals
- Fail Fast
- Diagnostics are Important

Crafting Good API's

- Essence vs. Ceremony
- The Pit of Success
- Consistency
- Predictability
- Discoverability
- Readability / Understandability

Takeaways

- Keep your code clean
- Make sure your abstractions match reality, and reality changes over time
- Minimize ceremony
- Test behaviors, not implementation details
- Good tests are readable tests