

C# Extension Methods

Module 4: Extension Method Library (part 1)

Elton Stoneman
geekswithblogs.net/eltonstoneman
elton@sixeyed.com



pluralsight
hardcore developer training

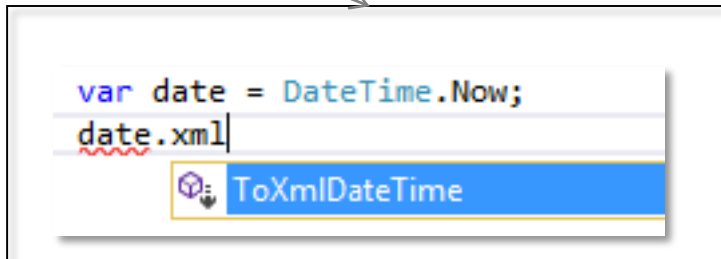
Extension Method Library

- Extension methods for every project

- Almost
- Reusable code & techniques

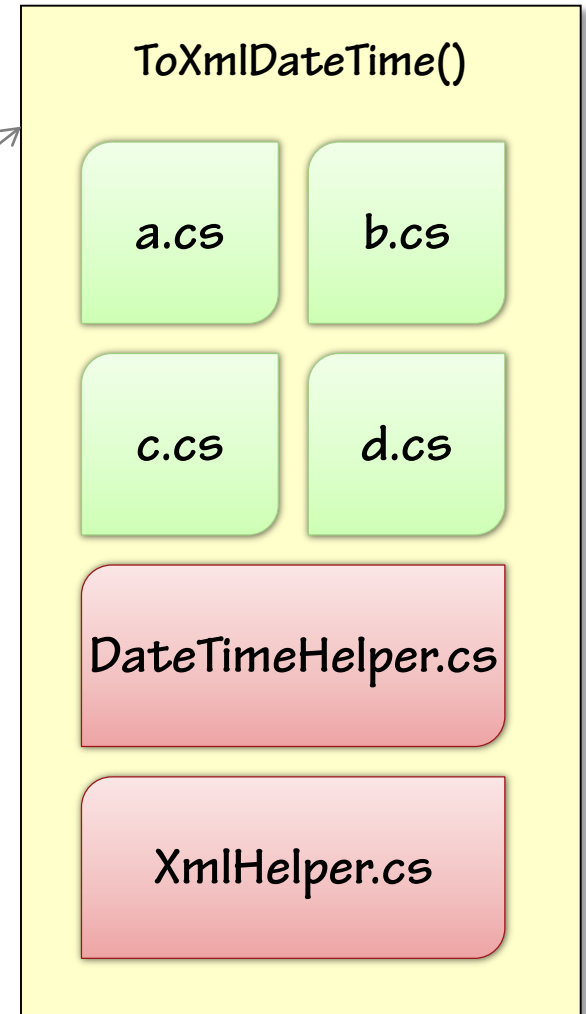
- Framework libraries

- Common code
- Visibility & avoiding duplication



```
var date = DateTime.Now;  
date.xml|  
ToXmlDateTime
```

The screenshot shows a code editor with the line `date.xml|`. A dropdown menu is open, showing the `ToXmlDateTime` extension method as a suggestion. An arrow points from the text 'Visibility & avoiding duplication' in the list above to this dropdown menu.



Extension Method Library

- **Extension methods for every project**
 - Almost
 - Reusable code & techniques
- **Part 1 – internals**
 - Core: exceptions, enums
 - Reflection & expressions
 - Entity Framework: easy auditing
- **Part 2 – externals**
 - WCF: safely closing clients
 - WebApi: location headers
 - MVC: HTML helpers

Extension Method Library

- **Extension methods for every project**

- Almost
- Reusable code & techniques

- **Part 1 – internals**

- Core: exceptions, enums
- Reflection & expressions
- Entity Framework: easy auditing

- **Part 2 – externals**

- WCF: safely closing clients
- WebApi: location headers
- MVC: HTML helpers



Demo 1: Core Framework

Feature

Extend core .NET
functionality to
add readable
output

Task

Extend Exception
to provide a full
message string

Task

Extend Enum to
output friendly
names

Demo 1: Core Framework

Demo 1: Core Framework

- **ExceptionExtensions**
 - In System namespace
 - Added `FullMessage()`

```
public static string FullMessage(this Exception ex)
```

```
try { Divide(10, 0); }  
catch(Exception ex)  
{  
    Debug.WriteLine(ex.FullMessage());  
}
```

Divide failed – amount: 10, by: 0
Invalid operation
Attempt to divide by zero

Demo 1: Core Framework

■ EnumExtensions

- In System namespace
- Added *GetName()* and *GetDescription()*

```
public static string GetName(this Enum enumValue)  
public static string GetDescription(this Enum enumValue)
```

```
Module.Intro.GetName() //"Intro"  
Module.Intro.GetDescription() //"Introducing Extension Methods"
```

■ Extended base class

- Methods available to new implementations
 - Custom Exception classes
 - New Enum definitions

Reflection & Expressions

■ Reflection with strings

- Access members by name
- No compile-time checking
- When definitions change, code breaks at runtime

■ Reflection with expressions

- Access members by accessing members
- Compile-time checking
- Accessible members only

```
var user = new User() { FirstName = "Elton" };  
var reader = new ObjectReader<User>(user);  
Assert.AreEqual("Elton", reader.GetValue<string>("FirstName")); //1
```

```
Assert.AreEqual("Elton", reader.GetValue(x => x.FirstName)); //2
```

Demo 2: Reflection & Expressions

Feature

Map between
classes
consistently and
safely

Task

Replace string
reflection with
expression
reflection

Demo 2: Reflection & Expressions

Demo 2: Reflection & Expressions

■ LambdaExpressionExtensions

- In System.Linq.Expressions namespace
- Add ToPropertyInfo()

```
public static PropertyInfo ToPropertyInfo  
                             (this LambdaExpression expression)
```

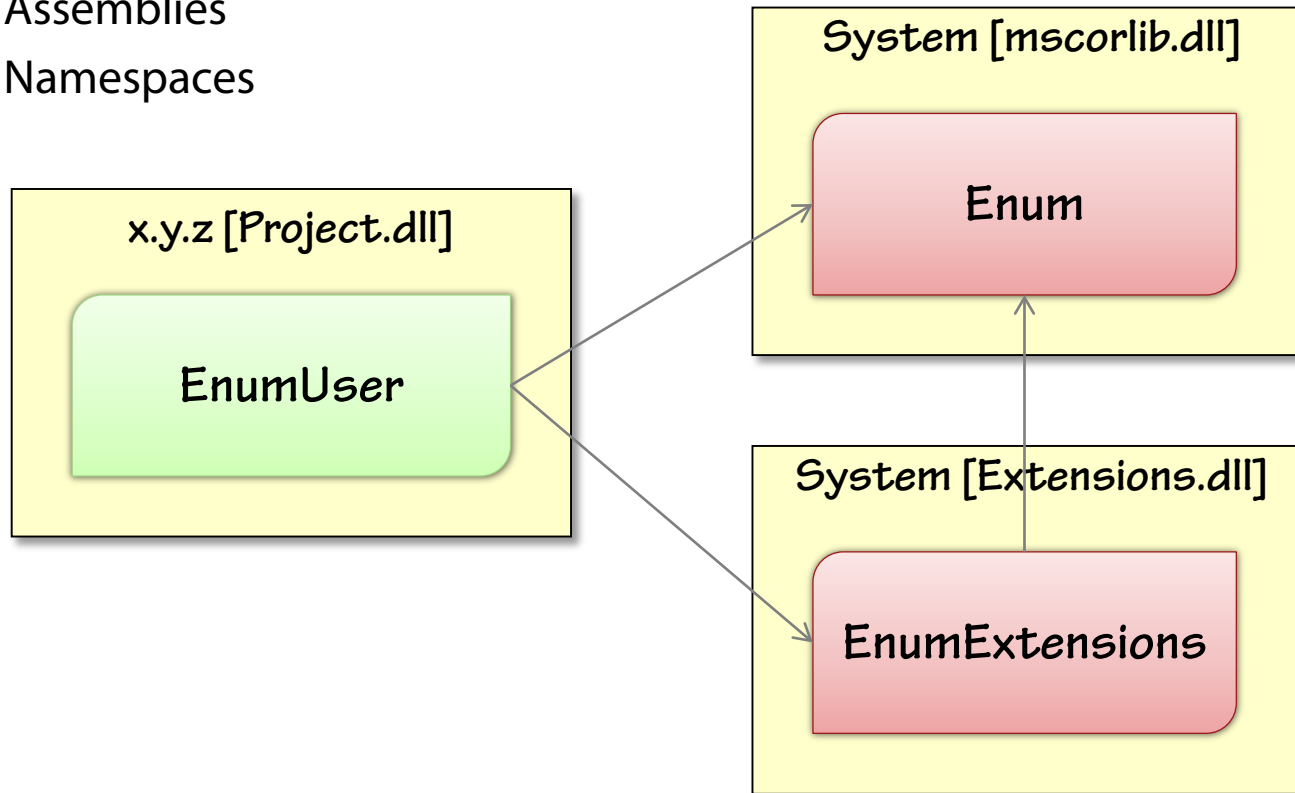
```
public Map<TSource, TTarget> Populate<T>(  
    Expression<Func<TTarget, T>> targetPropertyExpression,  
    Func<TSource, T> sourceValue)  
{  
    var accessor = targetPropertyExpression.ToPropertyInfo();
```

```
map.Populate(t => t.Name, s => s.FirstName + " " + t.LastName)
```

Entity Framework

- Structuring extension methods

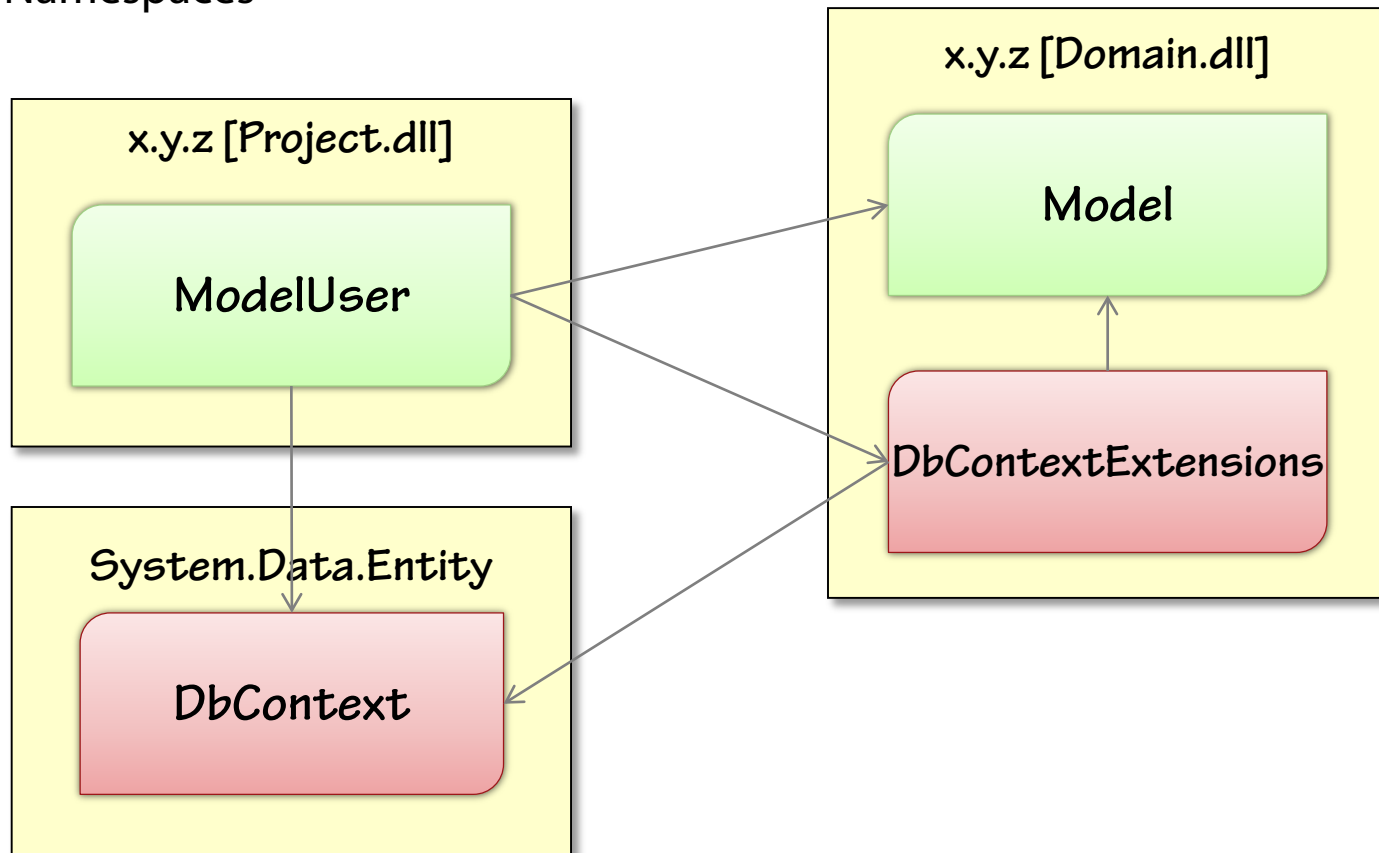
- Assemblies
- Namespaces



Entity Framework

- Structuring extension methods

- Assemblies
- Namespaces



Demo 3: Entity Framework

Feature

Add compliance requirements to database

Task

Audit changes with username and timestamp

Task

Ensure read-only reference data is not updated

Demo 3: Entity Framework

Demo 3: Entity Framework

■ DbContextExtensions

- Custom namespace – requires custom interfaces
- Added *Save()*

```
public static void Save(this DbContext context)
```

- Interrogate changed entities

```
foreach (var changedEntity in context.ChangeTracker.Entries()  
        .Where(x=>x.State == EntityState.Added ||  
                x.State == EntityState.Modified))
```

- Condition when *IReadOnly* or *IAudited*

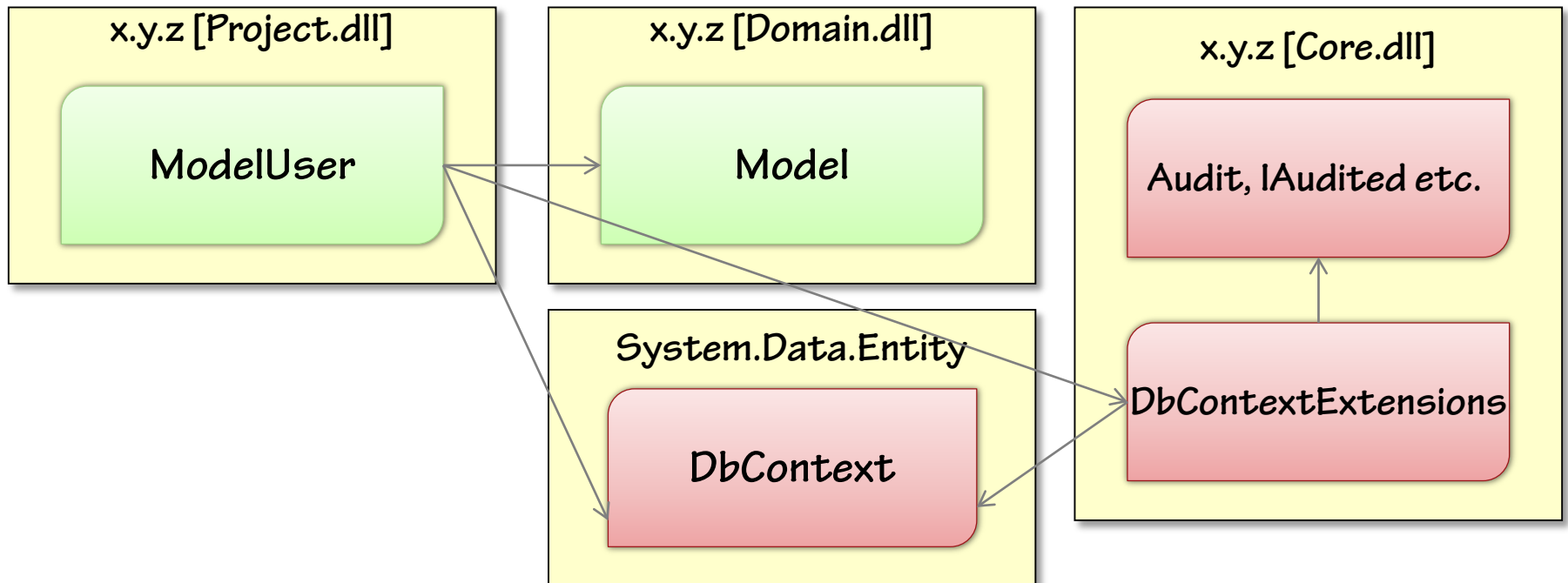
```
var readOnly = changedEntity.Entity as IReadOnly;  
//...  
var audited = changedEntity.Entity as IAudited;
```

- Then *SaveChanges()*

Demo 3: Entity Framework

■ DbContextExtensions

- *Save()* can be used with any EF model
 - If dependencies defined in framework library



- But *SaveChanges()* still available
 - Needs overriding in each DbContext class

Extension Method Library

- **Extension methods for every project**
 - Reusable code & techniques
- **More reasons to use extension methods**
 - Visibility of common features
 - Code safety & maintenance
- **Part 1 – internals**
 - Core: exceptions, enums
 - Reflection & expressions
 - Entity Framework: easy auditing
- **Part 2 – externals**
 - WCF: safely closing clients
 - WebApi: location headers
 - MVC: HTML helpers

Extension Method Library

- **Extension methods for every project**
 - Reusable code & techniques
- **More reasons to use extension methods**
 - Visibility of common features
 - Code safety & maintenance
- **Part 1 – internals**
 - Core: exceptions, enums
 - Reflection & expressions
 - Entity Framework: easy auditing
- **Part 2 – externals**
 - WCF: safely closing clients
 - WebApi: location headers
 - MVC: HTML helpers

