wisetech global

Advancing human potential.

UTS Guest Lecture: EF & Avoiding Bugs

Benjamin Sutas & Brenton Smith
Senior Software Engineers

17 October 2022

Meet Ben

Work

- 2009-2012 - USyd BIT (Hons)
- 2013-2017 - Optiver Australia
- 2017-Now – WiseTech Global
  - Core Team – 2 years
  - HRM Team – 3 years

Other

•2021-Now – developer on OpenLoco

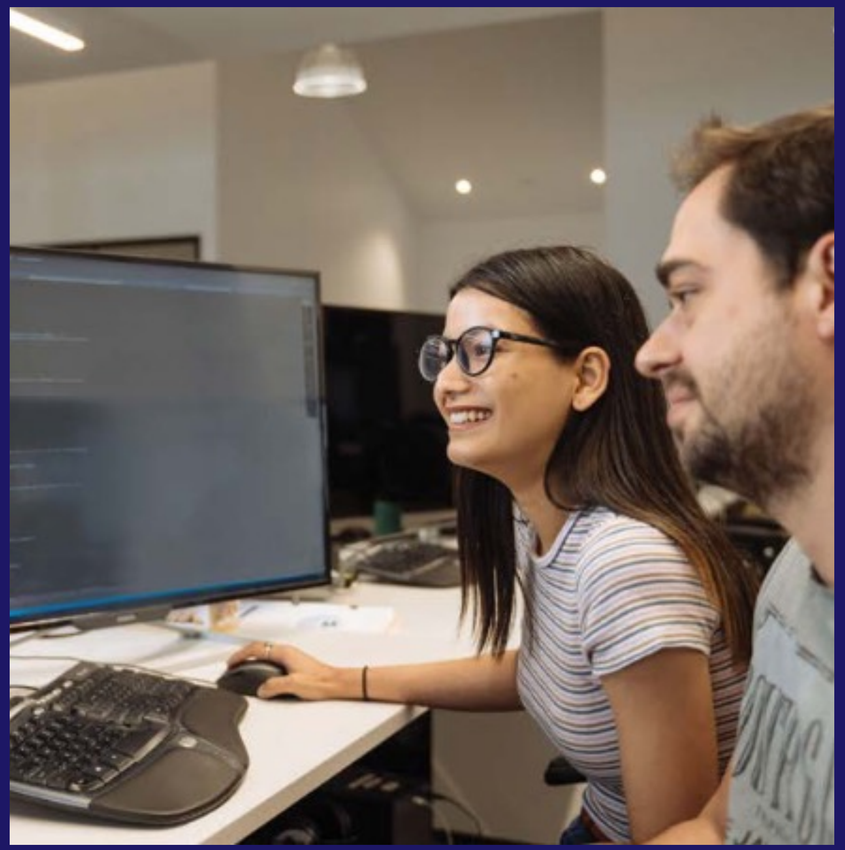
Meet Brenton

- 2012-2018: UTS B ICT Engineering/B Business
  - Programmers' Society President (2018)
- 2019-2022: Georgia Tech MSc in Comp Sci
- 2013-Now: WiseTech Global.
  - Warehouse team.

Where We Work

What is WiseTech Global?

- Vision to build the operating system for global logistics.
  - 41 of the top 50 global 3PL providers, 24 of the top 25 largest global freight forwarders.
- Headquartered in Sydney, ~2000 employees globally.
- Primarily .NET and C# development.
  - 15+ million lines of code, hundreds of developers.
  - Desktop, web and mobile applications.



wisetech global

Entity Framework (EF)

.NET Object Relational Mapper (ORM)

wisetech global

The Problem

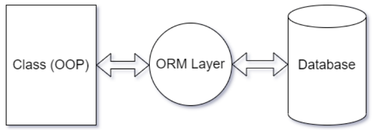
Data persistence

- You write a program
- You want to persist results of the program after program is terminated
- How to store?
  - File on computer has problems:
    - Access
    - Redundancy
    - Scale
    - Analytics/queries
    - Data model/schema
  - DB
- How program interact with DB?
  - Manual SQL queries, string parse results
  - Program can write queries for us

The Solution

ORM

- Maps database tables to code objects
- Hides raw queries with objects
- Compile time type safety



Notable .NET Implementations

- EF
- EF Core
- Dapper
- NHibernate
- ADO.NET

Features \ Framework	EF	EF Core	Dapper	NHibernate	ADO.NET
Change tracking	✓	✓	✗	✓	✗
Lazy Loading	✓	✓	✗	✓	✗
Supports different database providers	✓	✓	✓	✓	✓
Code First approach (Design DB from code)	✓	✓	✗	✗	✗
Caching	✓	✓	✗	✓	✗
Asynchronous operations	✓	✓	✓	✓	✓
Batch processing (DML)	✗	✓	✗	✓	✓
Works with stored procedures	✓	✓	✓	✓	✓

Entity Framework Core

- A modern, open-source, cross-platform .NET ORM that is the spiritual successor to the popular Entity Framework

- EF 1 – 2008
- EF Core 1.0 - 2016
- EF 6.4 - 2019
- EF Core 7.0 - Nov 2022

- Built with .NET 6.0
- Supports DB-first, model-first or code-first
- Can use LINQ to perform fluent-api style queries in an OO manner

How does it work

- EF Core library processes the LINQ query to build an internal query representation
  - This processing is cached
- This query is passes to a DB provider
  - The DB provider can identify which parts require DB queries, or which parts can be resolved using caches
  - Provider then translates any DB queries into the DB-specific language
  - Provider sends the query, waits for results
- From the DB provider, for each result EF Core will check if an entity (ie an object in your code) already exists for that result, and if it does it'll return that object, otherwise making a new object, adding it to the tracking list, and returning it
- This is all done lazily – when you create and run the LINQ statement, you are just constructing a query in memory. The query itself is only run when the results must be consumed, eg evaluating the list of results

Creating a model and context

- A DbContext instance represents a session with the database and can be used to query and save instances of your entities.
- DbContext is a combination of the Unit Of Work and Repository patterns.
- In this example, a DB file is made in our %appdata%/local folder
- In practice, this data source can be to a db server, in-memory db, a file, etc.

```
5 references
public class BloggingContext : DbContext
{
    3 references
    public DbSet<Blog> Blogs { get; set; }
    0 references
    public DbSet<Post> Posts { get; set; }

    2 references
    public string DbPath { get; }

    1 reference
    public BloggingContext()
    {
        var path = Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData);
        DbPath = Path.Join(path, "blogging.db");
    }

    0 references
    protected override void OnConfiguring(DbContextOptionsBuilder options)
        => options.UseSqlite($"Data Source={DbPath}");
}

[Table("tbl_blogs")]
3 references
public class Blog
{
    [Key]
    2 references
    public Guid BlogId { get; set; }

    [Url]
    2 references
    public string Url { get; set; }

    2 references
    public List<Post> Posts { get; } = new();
}

[Table("tbl_blog_posts")]
3 references
public class Post
{
    [Key]
    0 references
    public Guid PostId { get; set; }

    [Required]
    [StringLength(160)]
    1 reference
    public string Title { get; set; }
    1 reference
    public string Content { get; set; }

    [ForeignKey("Blog")]
    0 references
    public int BlogId { get; set; }
    0 references
    public Blog Blog { get; set; }
}
```

Change Tracking

Tracking your changes

- DbContext tracks changes made to entities
- These tracked entity changes dictate what happens to the database when SaveChanges() is called
- Entities become tracked when they're returned from a query, added to the DbContext, or connected to an existing tracked entity.
- Entities are no longer tracked when the DbContext is disposed, changes are cleared, or the entities are explicitly removed

Entity State	Tracked by DbContext	Exists in DB	Properties modified	Action on SaveChanges()
Detached	No	–	–	–
Added	Yes	No	–	Insert
Unchanged	Yes	Yes	No	–
Modified	Yes	Yes	Yes	Update
Deleted	Yes	Yes	–	Delete

CRUD

The four basic operations of persistent storage

CRUD	SQL	HTTP (Rest)
Create	INSERT	POST
Read	SELECT	GET
Update	UPDATE	PUT
Delete	DELETE	DELETE

```
// Create
db.Add(new Blog { Url = "https://wisetechglobal.com/" });
db.SaveChanges();

// Read
var blog = db.Blogs
    .OrderBy(b => b.BlogId)
    .First();

// Update
blog.Url = "https://www.wisetechglobal.com/";
var post = new Post { Title = "Hello World", Content = "This is a blog post about EF Core." };
blog.Posts.Add(post);
db.SaveChanges();

// Delete
db.Remove(blog);
db.SaveChanges();
```


wisetech global

Data Annotations

Providing contextual information to EF Core

System.ComponentModel.DataAnnotations.Schema attributes

Attribute	Description
Table	The database table and/or schema that a class is mapped to.
Column	The database column that a property is mapped to.
ForeignKey	Specifies the property is used as a foreign key in a relationship.
DatabaseGenerated	Specifies how the database generates values for a property.
NotMapped	Applied to properties or classes that are to be excluded from database mapping.
InverseProperty	Specifies the inverse of a navigation property
ComplexType	Denotes that the class is a complex type. *Not currently implemented in EF Core.

System.ComponentModel.Annotations attributes

Attribute	Description
Key	Identifies one or more properties as a Key
Timestamp	Specifies the data type of the database column as `rowversion`
ConcurrencyCheck	Specifies that the property is included in concurrency checks
Required	Specifies that the property's value is required
MaxLength	Sets the maximum allowed length of the property value (string or array)
StringLength	Sets the maximum allowed length of the property value (string or array)

```
[Table("tbl_blogs")]
3 references
public class Blog
{
    [Key]
    2 references
    public Guid BlogId { get; set; }

    [Url]
    2 references
    public string Url { get; set; }

    2 references
    public List<Post> Posts { get; } = new();
}

[Table("tbl_blog_posts")]
3 references
public class Post
{
    [Key]
    0 references
    public Guid PostId { get; set; }

    [Required]
    [StringLength(160)]
    1 reference
    public string Title { get; set; }
    1 reference
    public string Content { get; set; }

    [ForeignKey("Blog")]
    0 references
    public int BlogId { get; set; }
    0 references
    public Blog Blog { get; set; }
}
```

Migrations

aka schema changes

- Tools
  - .NET CLI or PMC, MS recommend .NET CLI
- Be careful!

C# Copy

```
public class Blog
{
    public int Id { get; set; }
    public string Name { get; set; }
    public DateTime CreatedTimestamp { get; set; }
}
```

.NET CLI Copy

```
dotnet ef migrations add AddBlogCreatedTimestamp
```

.NET CLI Copy

```
dotnet ef database update
```


wisetech global

Migrations

aka schema changes

- Tools
  - .NET CLI or PMC, MS recommend .NET CLI
- Be careful!
  - Data can be lost – renaming columns – EF Core can handle

C# Copy

```
migrationBuilder.DropColumn(
    name: "Name",
    table: "Customers");

migrationBuilder.AddColumn<string>(
    name: "FullName",
    table: "Customers",
    nullable: true);
```

C# Copy

```
migrationBuilder.RenameColumn(
    name: "Name",
    table: "Customers",
    newName: "FullName");
```


wisetech global

Migrations

aka schema changes

- Tools
  - .NET CLI or PMC, MS recommend .NET CLI
- Be careful!
  - Data can be lost – renaming columns – EF Core can handle
  - Data can be lost – merging columns – EF Core cannot handle

C# Copy

```
migrationBuilder.DropColumn(
    name: "FirstName",
    table: "Customer");

migrationBuilder.DropColumn(
    name: "LastName",
    table: "Customer");

migrationBuilder.AddColumn<string>(
    name: "FullName",
    table: "Customer",
    nullable: true);
```

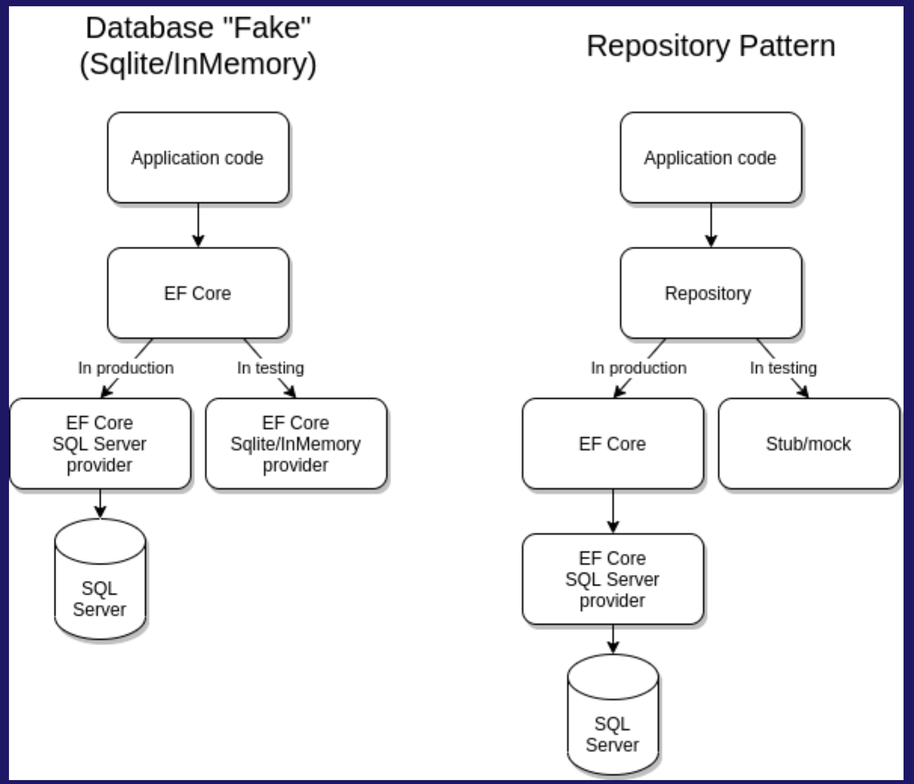
C# Copy

```
migrationBuilder.AddColumn<string>(
    name: "FullName",
    table: "Customer",
    nullable: true);

migrationBuilder.Sql(
@"
    UPDATE Customer
    SET FullName = FirstName + ' ' + LastName;
");

migrationBuilder.DropColumn(
    name: "FirstName",
    table: "Customer");

migrationBuilder.DropColumn(
    name: "LastName",
    table: "Customer");
```


wisetech global

Testing

- Against production/real DB or not
- In-memory provider
- SQLite in-memory provider
- Repository pattern



Testing

- Against production/real DB or not
- In-memory provider
- SQLite in-memory provider
- Repository pattern

Feature	In-memory	SQLite in-memory	Mock DbContext	Repository pattern	Testing against the database
Test double type	Fake	Fake	Fake	Mock/stub	Real, no double
Raw SQL?	No	Depends	No	Yes	Yes
Transactions?	No (ignored)	Yes	Yes	Yes	Yes
Provider-specific translations?	No	No	No	Yes	Yes
Exact query behavior?	Depends	Depends	Depends	Yes	Yes
Can use LINQ anywhere in the application?	Yes	Yes	Yes	No*	Yes

Miscellaneous

- Can still use raw SQL if absolutely necessary

```
var blogs = db.Blogs
    .FromSqlRaw($"SELECT * FROM dbo.Blogs")
    .ToList();
```

- SQL Injection is still possible if you aren't careful (this specific one isn't possible in EF Core 7.0)

```
var user = "johndoe";
var blogs1 = db.Blogs
    .FromSqlRaw($"EXECUTE dbo.GetMostPopularBlogsForUser {user}")
    .ToList();
```

- Pagination
- Logging
- Database functions / stored procedures
- async

wisetech global

Links

- Overview – https://learn.microsoft.com/en-us/ef/core/
- Source – https://github.com/dotnet/efcore
- EFCore tutorial – https://learn.microsoft.com/en-us/ef/core/get-started/overview/first-app?tabs=visual-studio
- EFCore PMC tools – https://learn.microsoft.com/en-us/ef/core/cli/powershell
- Comparison in code – https://www.tatvasoft.com/blog/what-are-orms-and-how-does-it-work/
- Follow the ASP.NET Core Tutorial to use EF Core in a web app
- Learn more about LINQ query expressions

Avoiding Bugs

Techniques WTG Use To Avoid Bugs

wisetech global

Why Defects Matter

- Modern businesses cannot operate without their software.
  - Enterprise software is therefore near mission critical!

- Unexpected effort to provide support and address defects.
  - Less time to add value.

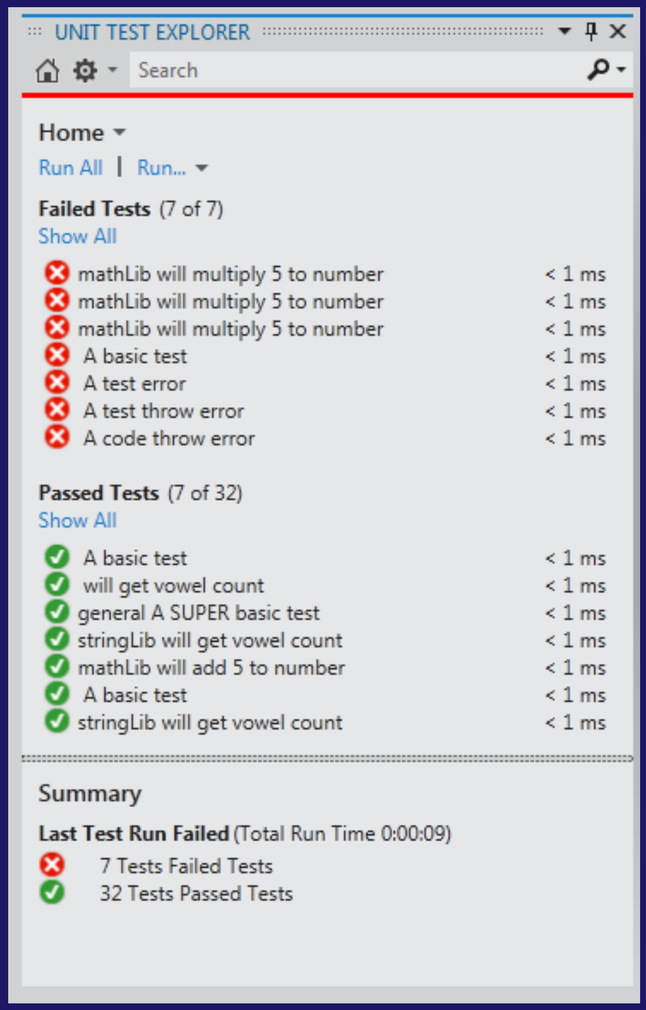
- (Fix) “Defects First” approach.

How To Minimise Bugs

- Compiler checks and static analysis.
  - Type systems, compiler errors/warnings.
  - Linting and analysers for code smells.
- Code reviews/walkthroughs.
- User Acceptance Testing (functional reviews).
- Release rings/branches.
- ***Automated unit testing.***
- ***Continuous Integration.***

Unit Testing

- Write code to test a small part of your code.
- CargoWise One has ~1.2m automated tests.
- .NET has NUnit, xUnit.net, MSTest frameworks to facilitate writing tests.
  - Integrate with Visual Studio/Rider/VSCode.



Arrange/Act/Assert

The three A's of Unit Testing.

```
[Test]
0 references | 0 changes | 0 authors, 0 changes
public void MoveRightIncrementsXValue()
{
    // Arrange
    // Create a new block
    var block = new Block();

    // (Pre) Assert
    // Expect the block to start at 0,0
    Assert.That(block.X, Is.Zero);
    Assert.That(block.Y, Is.Zero);

    // Act
    // Move to the right
    block.Move(Direction.Right);

    // Assert
    // Only the X value should be modified by horizontal movement
    Assert.That(block.X, Is.EqualTo(1));
    Assert.That(block.Y, Is.Zero);
}
```

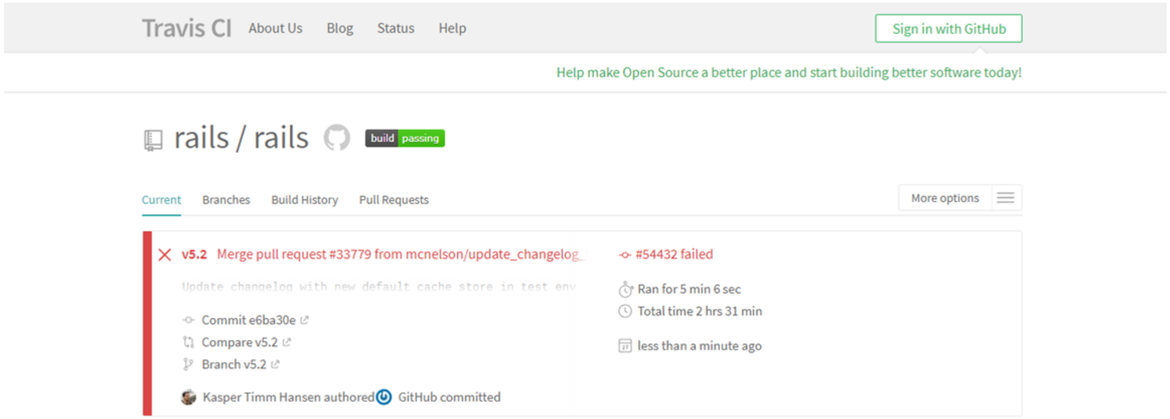

wisetech global

Test-Driven Development (TDD)

- At WiseTech we write tests before code!
  - Ever used auto-marking systems?
- TDD encourages good object-oriented design.
  - SOLID principles.
- Dependency Injection and mocking.

Continuous Integration

- Ever done a group assignment…?
- **Gate merging to master:** merge, build, static analysis, tests must pass.
- WTG has DAT (proprietary), but you can do this yourself!
  - Azure DevOps
  - Jenkins (open source)
  - CircleCI
  - Travis CI (Free in GitHub student pack)
  - GitLab
  - TeamCity
  - DIY with Git hooks



Career Advice

wisetech global

Career Advice

- Network for any opportunities: internships, startups, projects.
- Build a portfolio to show to recruiters/interviewers!
- Practice data structures and algorithms.
  - **Competitive Programming:** UTS ProgSoc, ICPC, Google CodeJam, Codeforces.
  - LeetCode
- Software Engineering principles > learning another language.
  - OO, SOLID principles, Design Patterns.
  - DevOps: Git, unit testing.

Thanks for listening!

Any questions?

wisetech global