

Automated Testing with EF Core



Julie Lerman

MOST TRUSTED AUTHORITY ON ENTITY FRAMEWORK

@julielerman thedatafarm.com



Manual Testing

```
private static void ReplaceAHorse()
{
    //var samurai = _context.Samurais.Include(s => s.Horse).FirstOrDefault(s => s.Id == 23);
    var samurai = _context.Samurais.Find(23); //has a horse
    samurai.Horse = new Horse { Name = "Trigger" };
    _context.SaveChanges();
}
```

Exception Unhandled

SqlException: Cannot insert duplicate key row in object 'dbo.Horses' with unique index 'IX_Horses_SamuraiId'. The duplicate key value is (23).

```
Executed DbCommand (23ms) [Parameters=[], CommandType='Text', CommandTimeout='30']
DELETE FROM [Horses]
WHERE [Id] = @p0;
SET NOCOUNT ON;
SELECT @@ROWCOUNT;
Executed DbCommand (2ms) [Parameters=[23], CommandType='Text', CommandTimeout='30']
SELECT [Id]
FROM [Horses]
WHERE [Id] = 23
SET NOCOUNT = 1 AND [
```

```
[
  {
    id: 1,
    name: "JulieSanSan",
    quotes: [ ],
    clan: null,
    samuraiBattles: null,
    horse: null
  },
  {
    id: 2,
    name: "SampsonSan",
    quotes: [ ],
    clan: null,
    samuraiBattles: null,
    horse: null
  },
  {
    id: 16,
    name: "TashaSan",
    quotes: [ ],
```

QuickWatch

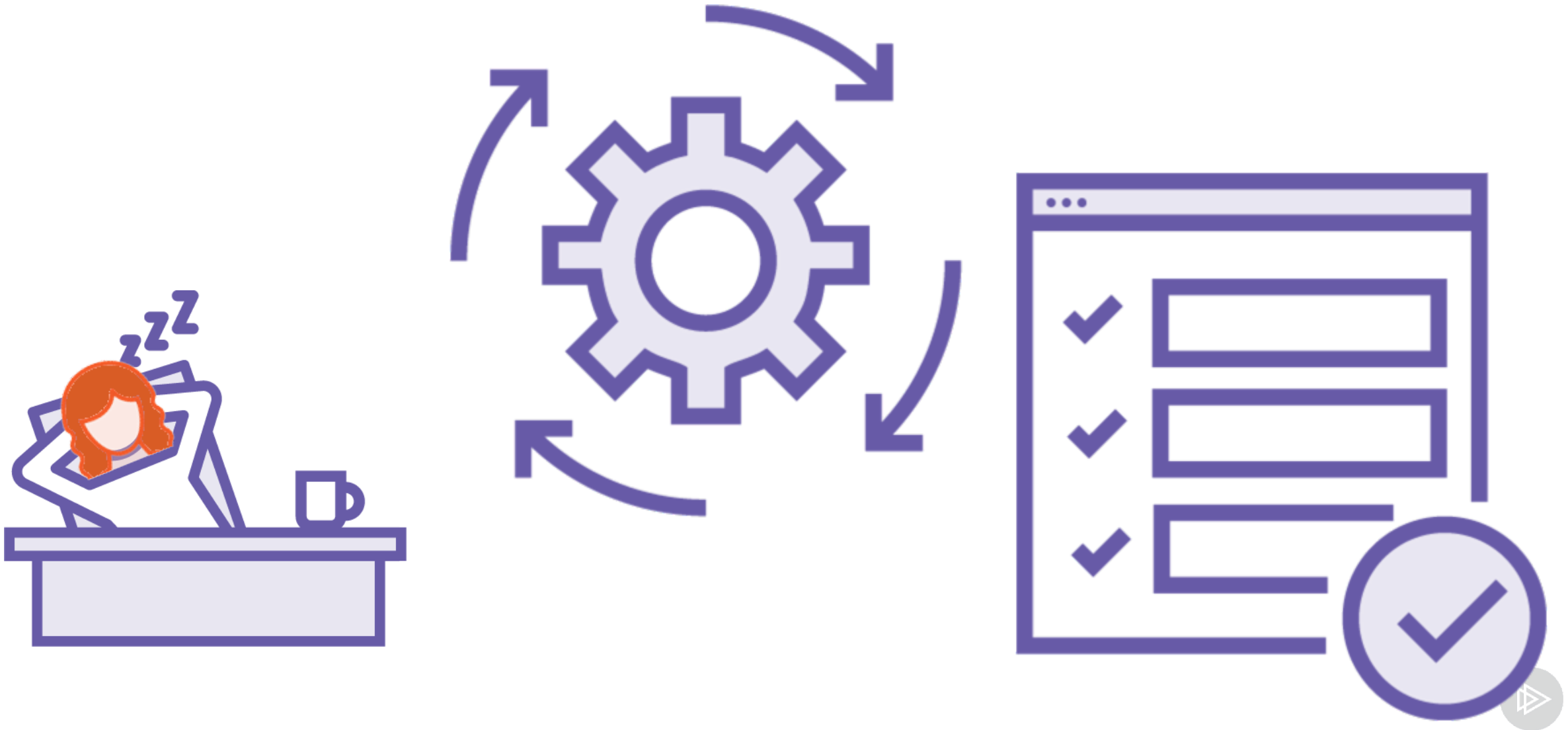
Expression: `_context.ChangeTracker.Entries().results`

Value:

Name	Value	Type
<code>_context.ChangeTracker.Entries()</code>	<code>{System.Linq.Enumerable.SelectEnumerableIterator<Microsoft.E...}</code>	<code>System.Collections....</code>
Current	<code>null</code>	<code>Microsoft.EntityFra...</code>
Non-Public members		
Results View	Expanding the Results View will enumerate the IEnumerable	
[0]	<code>{{Id: 1} Modified EntityType: Samurai}</code>	<code>Microsoft.EntityFra...</code>
[1]	<code>{{Id: 4} Unchanged EntityType: Quote}</code>	<code>Microsoft.EntityFra...</code>
[2]	<code>{{Id: 2} Unchanged EntityType: Samurai}</code>	<code>Microsoft.EntityFra...</code>
[3]	<code>{{Id: 16} Unchanged EntityType: Samurai}</code>	<code>Microsoft.EntityFra...</code>
[4]	<code>{{Id: 17} Unchanged EntityType: Samurai}</code>	<code>Microsoft.EntityFra...</code>
[5]	<code>{{Id: 21} Unchanged EntityType: Samurai}</code>	<code>Microsoft.EntityFra...</code>
[6]	<code>{{Id: 22} Unchanged EntityType: Samurai}</code>	<code>Microsoft.EntityFra...</code>
[7]	<code>{{Id: 23} Unchanged EntityType: Samurai}</code>	<code>Microsoft.EntityFra...</code>
[8]	<code>{{Id: 30} Unchanged EntityType: Samurai}</code>	<code>Microsoft.EntityFra...</code>
[9]	<code>{{Id: 31} Unchanged EntityType: Samurai}</code>	<code>Microsoft.EntityFra...</code>



Automated Testing



Module Overview



Super quick testing intro

What does it mean to test EF Core?

Writing tests that use a database

Writing tests to use an in-memory database

Testing EF Core used in your apps

Testing EF Core when used in a web app

Refactor demos for testability

Learn about structure of testable apps



A Very Quick Testing Overview



Common Types of Automated Tests



Unit Test

Test small units of your own code

```
public class Person
{
    public Person(string first, string last)
    {
        FirstName = first;
        LastName = last;
    }
    public string FirstName { get; }
    public string LastName { get; }
}
```

```
var person = new Person("Maria", "Adigon");
```

```
Is person.FirstName=="Maria" ?
Is person.LastName=="Adigon" ?
```



Common Types of Automated Tests



Unit Test

Test small units of your own code



Integration Test

Test that your logic interacts with other services or modules



Functional Test

Verify results of interaction



Many other types of automated testing



Understanding What We Mean by “Testing EF Core”



Testing EF Core Directly or Indirectly



Validate your DbContext against the database



Validate your business logic against the DbContext



Validate your business logic that uses the DbContext and database



Creating Your First Test and Using It Against the Database



Exploring Test Results & Performance Considerations



When Your Database Server Is a Resource Hog



Use SQL Server

Need to test specific behaviors of the target database



SQLite or SQL CE

Need to test generic SQL database behaviors



EF Core's InMemory DB

Need to test EF Core behavior or biz logic that uses EF Core

Using the InMemory Provider in Place of a Database Provider



Microsoft.EntityFrameworkCore.InMemory



Emulates RDBMS via in-memory lists



Handles generic RDBMS scenarios



Great alternative for test mocks



Requires mods to our existing solution



Writing Your First Test with the InMemory Provider



InMemoryDatabase Connection String

```
public void Test1()  
{  
  
    builder  
        .UseInMemoryDatabase("Test1DB");  
}
```

Test1DB is a fresh, empty, unique in memory database

```
public void TestX()  
{  
    builder  
        .UseInMemoryDatabase("TestXYDB");  
}  
  
public void TestY()  
{  
    builder  
        .UseInMemoryDatabase("TestXYDB");  
}
```

TestXYDB is a fresh database in the first method, and will get reused in the second



Refactoring and Testing Some Console App Logic



Re-using an InMemory Provider Database



Testing EF Core in an ASP.NET Core App

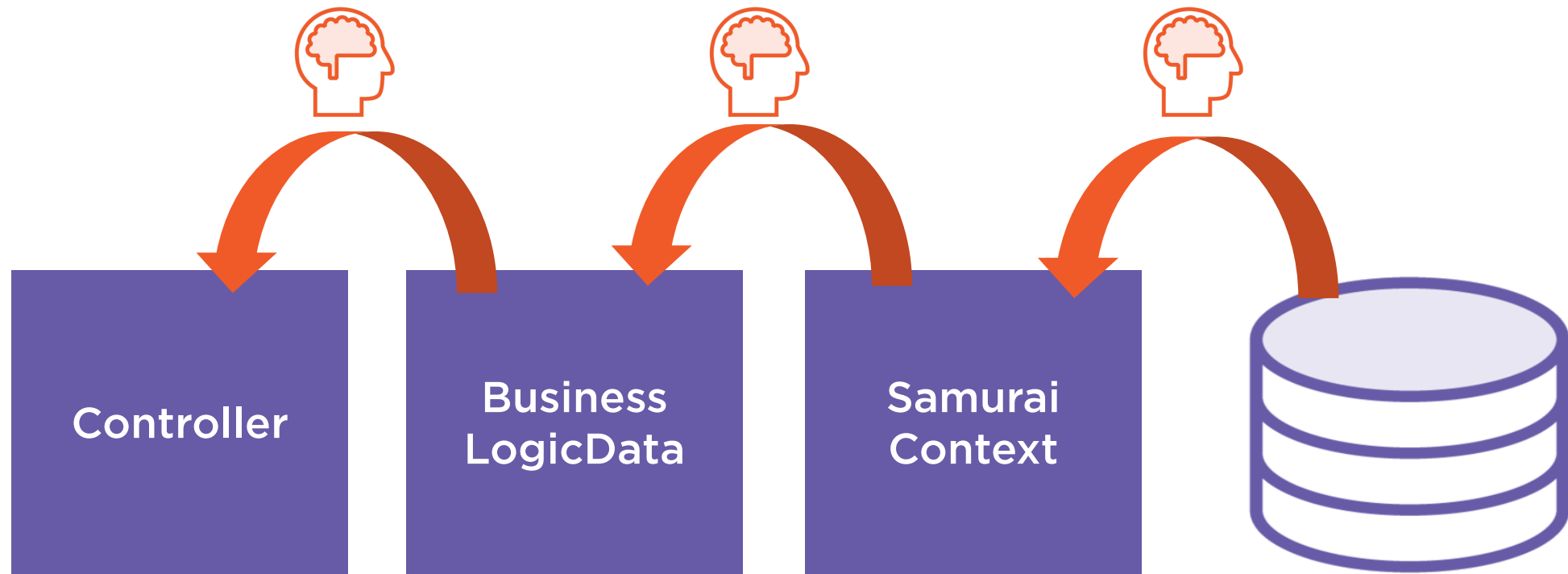


Testing the API's DbContext Configuration



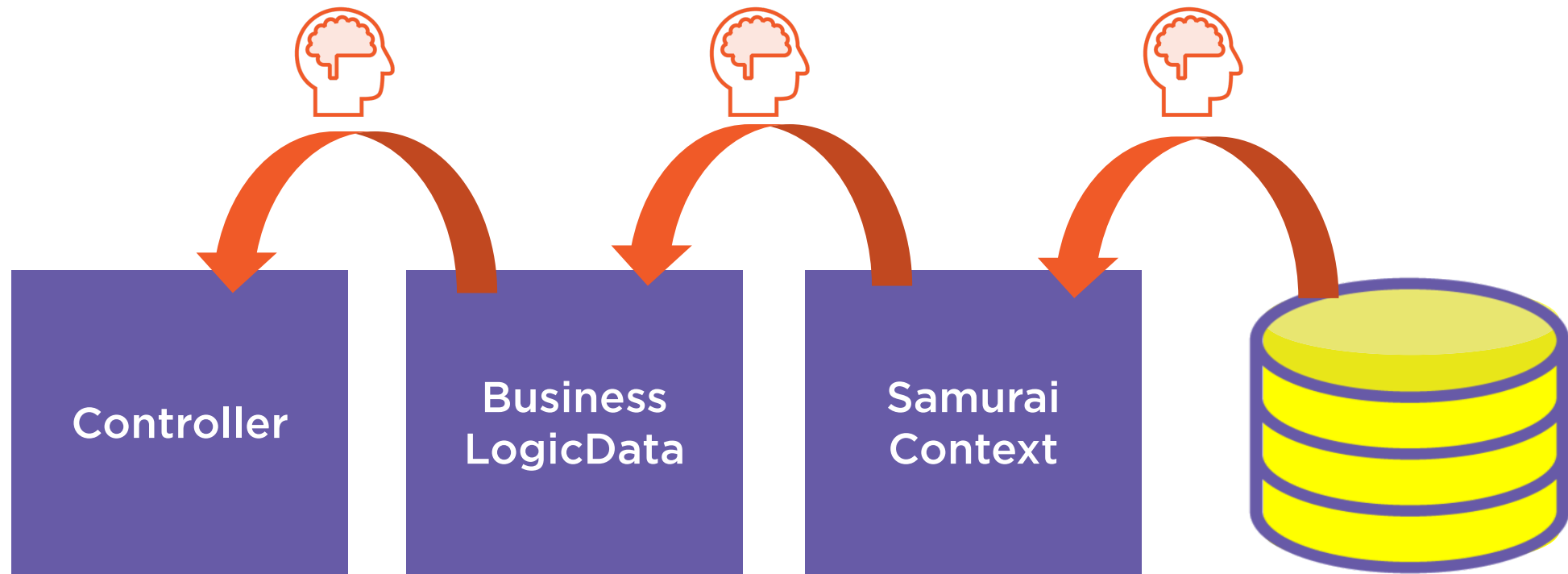
Deep Integration/Interaction Testing

API Configuration



Deep Integration/Interaction Testing

API Configuration



Review

Testing is an important skill

Verify that EF Core comprehends your DbContext mappings

Testing against the app's database is not always necessary or efficient

Substitute with lighter db or InMemory provider

Test only logic that uses EF Core

Test full integration of app with EF Core

Resources

Entity Framework Core on GitHub github.com/aspnet/entityframework

EF Core Documentation docs.microsoft.com/ef

Microsoft Docs on ASP.NET Core Testing

docs.microsoft.com/en-us/aspnet/core/test/integration-tests?view=aspnetcore-3.1

Course: Automated Testing for Fraidy Cats Like Me bit.ly/PS_Testing



Entity Framework Core: Getting Started



Julie Lerman

MOST TRUSTED AUTHORITY ON ENTITY FRAMEWORK

@julielerman thedatafarm.com

