

Mapping and Using Many-to-many Relationships



Julie Lerman

MOST TRUSTED AUTHORITY ON ENTITY FRAMEWORK

@julielerman thedatafarm.com



Module Overview



Review existing many-to-many relationship

Understand how to define a many-to-many that EF Core will comprehend

Creating, retrieving and modifying data in many-to-many graphs

Learn how change tracker and whether or not objects are in memory affect behavior



Reviewing the Existing Many-to-many Relationship and Mappings



Inserting Data in Many-to-many Relationships

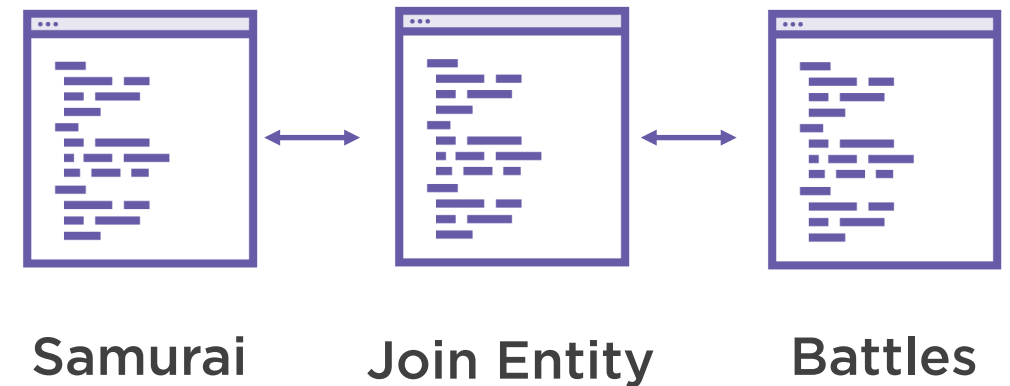


Many-to-many Relationship

Relational Database: Join Table




EF Core: Join Entity



```
public class Samurai
{
    public int Id { get; set; }
    public string Name { get; set; }
    public List<Battle> Battles { get; set; }
}

public class Battle
{
    public int Id { get; set; }
    public string Name { get; set; }
    public List<Samurai> Samurais { get; set; }
}
```



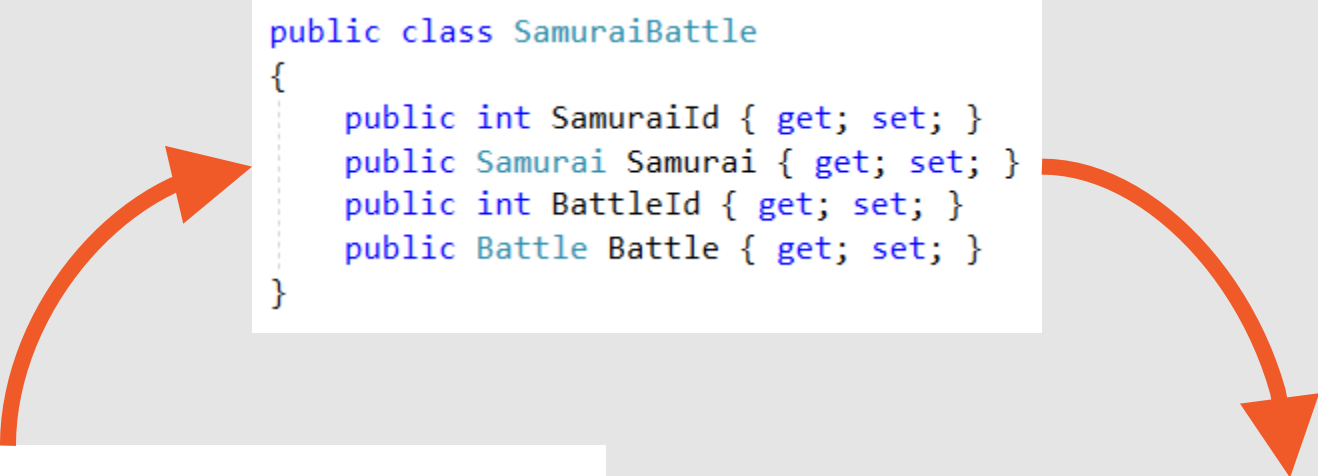
EF6 (& earlier) Understood Direct Many-to-many

EF Core does not (yet) comprehend this mapping



Join Entity with FKs and Navigation References

```
public class SamuraiBattle
{
    public int SamuraiId { get; set; }
    public Samurai Samurai { get; set; }
    public int BattleId { get; set; }
    public Battle Battle { get; set; }
}
```



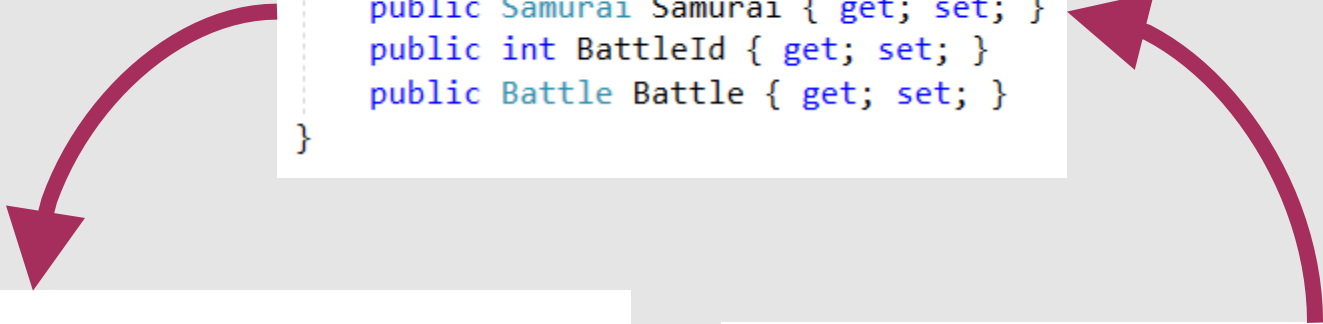
```
public class Samurai
{
    public Samurai()
    {
        Quotes = new List<Quote>();
        SamuraiBattles = new List<SamuraiBattle>();
    }
    public int Id { get; set; }
    public string Name { get; set; }
    public List<Quote> Quotes { get; set; }
    public List<SamuraiBattle> SamuraiBattles { get; set; }
    public SecretIdentity SecretIdentity { get; set; }
}
```

```
public class Battle
{
    public Battle()
    {
        SamuraiBattles = new List<SamuraiBattle>();
    }
    public int Id { get; set; }
    public string Name { get; set; }
    public DateTime StartDate { get; set; }
    public DateTime EndDate { get; set; }
    public List<SamuraiBattle> SamuraiBattles { get; set; }
}
```



Join Entity with FKs and Navigation References

```
public class SamuraiBattle
{
    public int SamuraiId { get; set; }
    public Samurai Samurai { get; set; }
    public int BattleId { get; set; }
    public Battle Battle { get; set; }
}
```



```
public class Samurai
{
    public Samurai()
    {
        Quotes = new List<Quote>();
        SamuraiBattles = new List<SamuraiBattle>();
    }
    public int Id { get; set; }
    public string Name { get; set; }
    public List<Quote> Quotes { get; set; }
    public List<SamuraiBattle> SamuraiBattles { get; set; }
    public SecretIdentity SecretIdentity { get; set; }
}
```

```
public class Battle
{
    public Battle()
    {
        SamuraiBattles = new List<SamuraiBattle>();
    }
    public int Id { get; set; }
    public string Name { get; set; }
    public DateTime StartDate { get; set; }
    public DateTime EndDate { get; set; }
    public List<SamuraiBattle> SamuraiBattles { get; set; }
}
```





SamuraiBattle
Kikuchiyo
+
Siege of Osaka



```
_context.Samurais.Add(samurai)
_context.Samurais.AddRange(samuraiList)
```

```
_context.Add(samurai)
_context.AddRange(samurai, battle)
```

```
_context.Samurais.Update(samurai)
_context.Samurais.UpdateRange(samuraiList)
```

```
_context.Update(samurai)
_context.UpdateRange(samurai, battle)
```

```
_context.Samurais.Remove(samurai)
_context.Samurais.RemoveRange(samuraiList)
```

```
_context.Remove(samurai)
_context.RemoveRange(samurai, battle)
```

◀ DbSet Add, AddRange

◀ DbContext Add, AddRange

◀ DbSet Update,
UpdateRange

◀ DbContext Update,
UpdateRange

◀ DbSet Remove,
RemoveRange

◀ DbContext Remove,
RemoveRange



Untracked Graph Behavior



Adding
New Children
to Pre-existing,
Unmodified Parent

DbSet.Add(Parent)

*All objects marked Added
All objects inserted into DB*

DbSet.Attach(Parent)

*Default: all objects marked Unchanged
Objects with no key value, marked Added
Added objects are inserted into DB*

Untracked Graph Behavior



Adding
New Children
to Pre-existing,
Edited Parent

DbSet.Add(Parent)

All objects marked Added
All objects inserted into DB

DbSet.Update(Parent)

Default: all objects marked Modified

Objects with no key value, marked Added

Modified objects are updated in DB

Added objects are inserted into DB



Adding Many-to-many Ends on the Fly

Existing Samurai

**Child
SamuraiBattle**

**Grandchild
New Battle**

Existing Battle

**Child
SamuraiBattle**

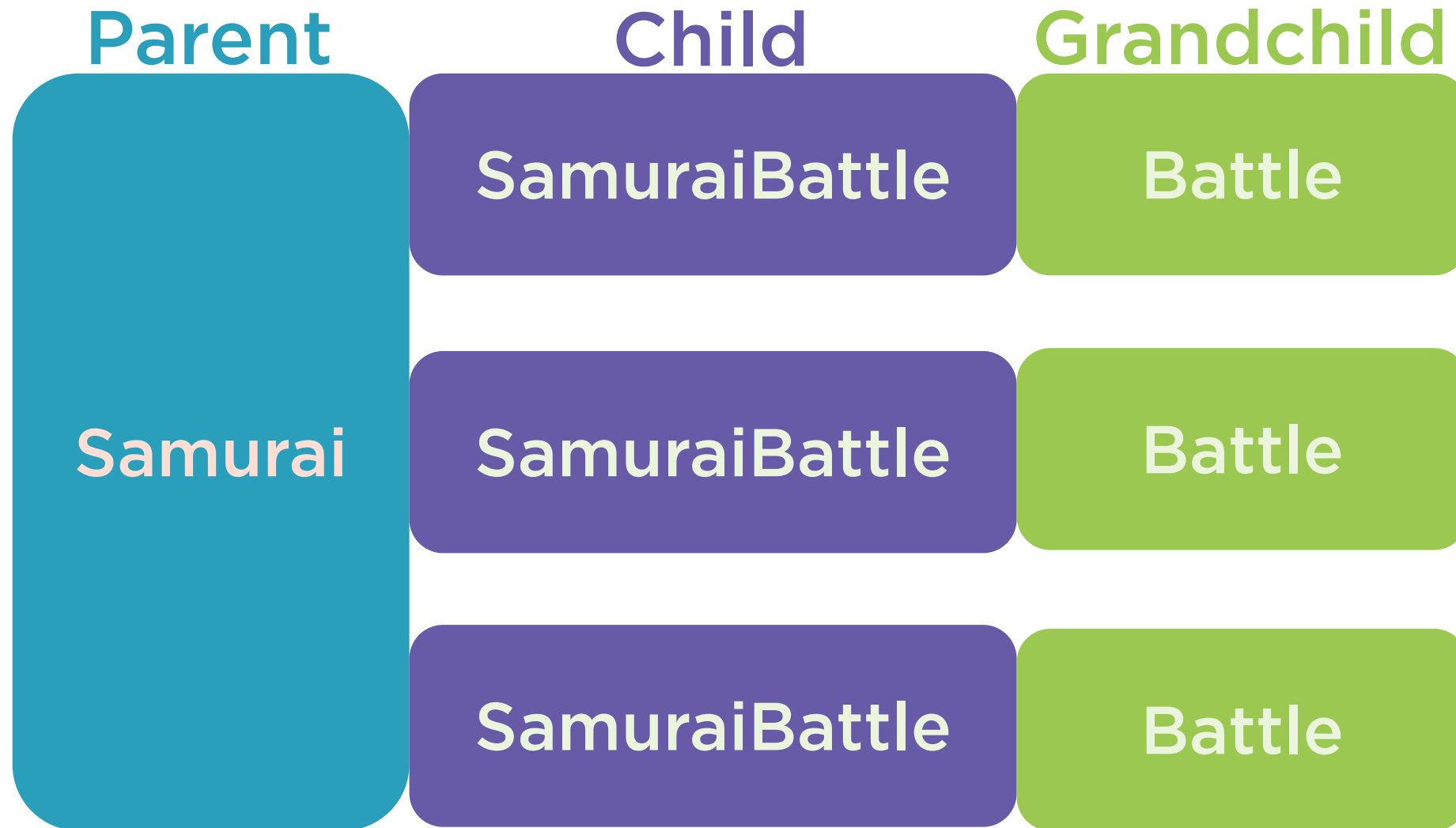
**Grandchild
New Samurai**



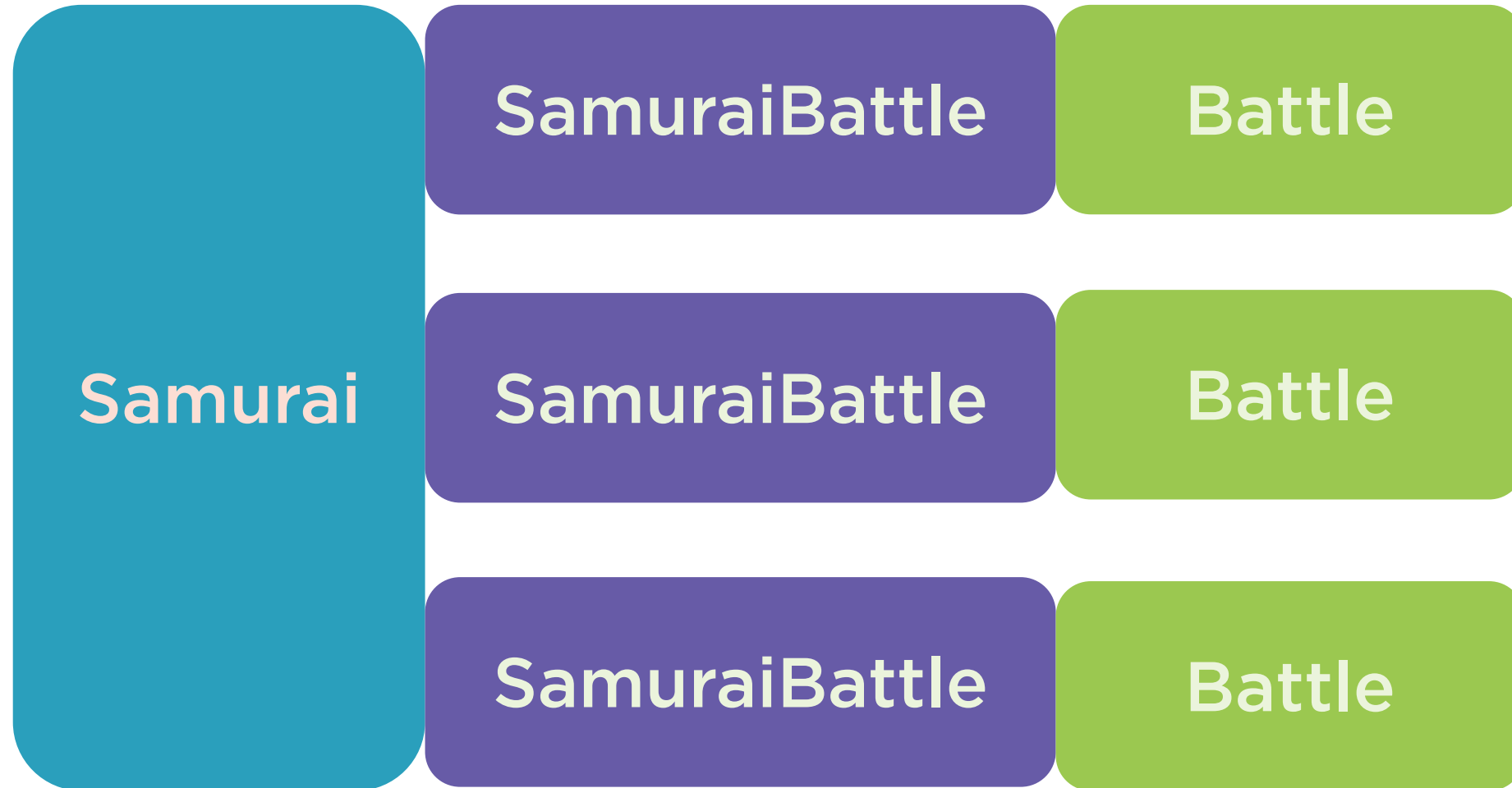
Retrieving Many-to-many Related Data



Many-to-many: Just Children & Grandchildren



Many-to-many: Just Children & Grandchildren




```
var battle = _context.Battles.Find(1);  
  
_context.Entry(battle)  
    .Collection(b => b.SamuraiBattles)  
    .Query()  
    .Include(sb => sb.Samurai)  
    .Load();
```

Explicit Loading When “Parent” Is in Memory

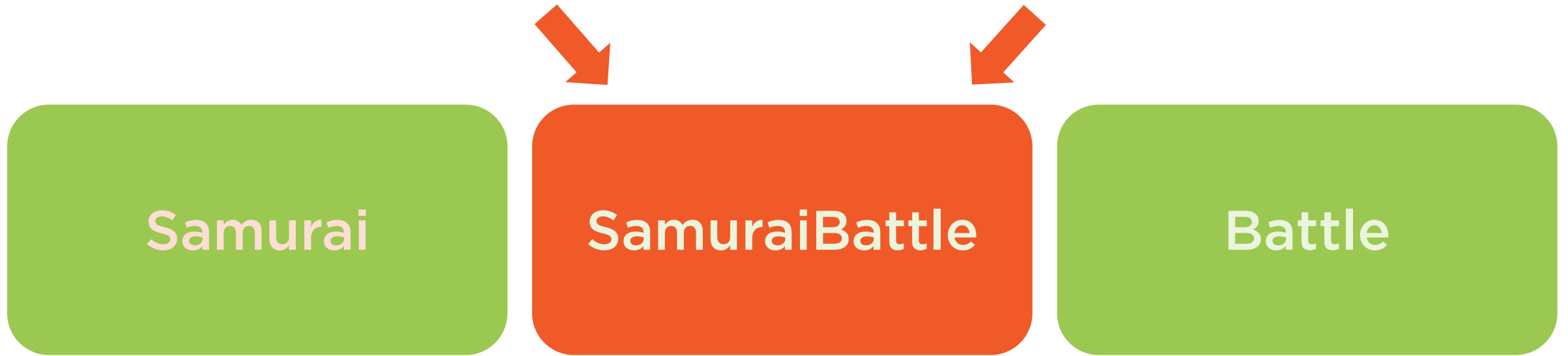
Coming up in a later module of this course



Modifying Many-to-many Relationships



Modifying or Deleting the Join, Not the Ends



Modifying the Join

Samurai

Id=1

SamuraiBattle

SamuraiId=1

BattleId=2

Battle

Id=2



Replacing the Join aka Delete a Join, Then Add a Join

Samurai

Id=1

SamuraiBattle

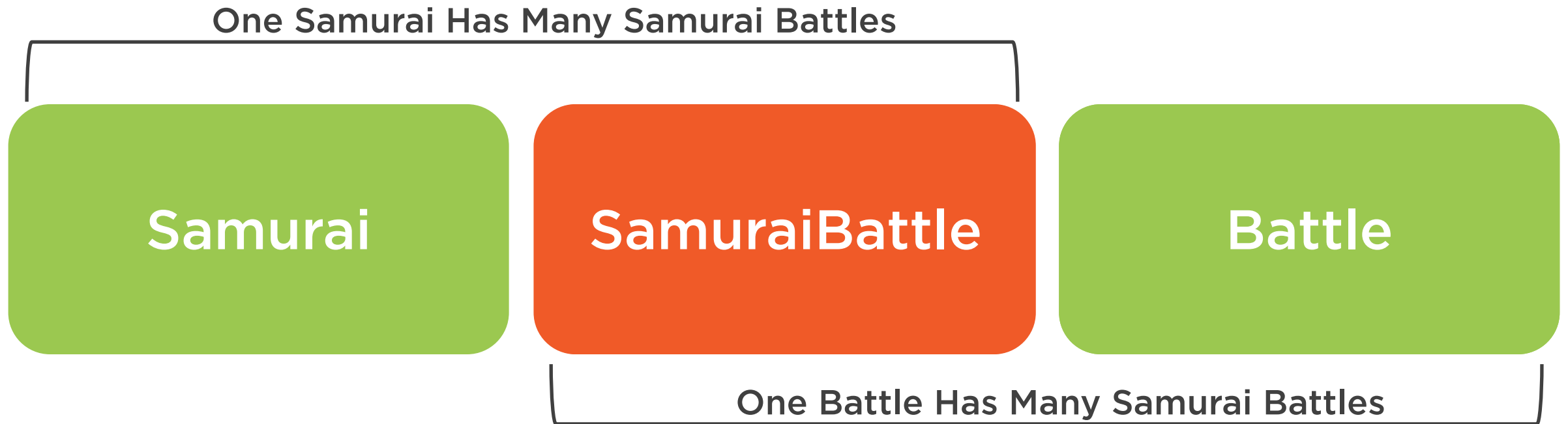
SamuraiId=1
BattleId=2

Battle

Id=2



Many-to-many is a Pair of One-to-Many



Review

EF Core currently only supports an explicit many-to-many mapping

Lots of scenarios to consider for creating or modifying a many-to-many relationship

Behavior is affected by whether or not the objects are in memory

Also affected by whether or not the objects are being tracked

Simplifying business logic

Resources

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

EF Core Roadmap bit.ly/efcoreroadmap

EF Core Documentation docs.microsoft.com/ef

[Pluralsight] Entity Framework Core 2: Getting Started bit.ly/PS_EFCore2

Starting point of Arthur Vickers blog series on many-to-many:

blog.oneunicorn.com/2017/09/25/many-to-many-relationships-in-ef-core-2-0-part-1-the-basics/



Mapping and Using Many-to-many Relationships



Julie Lerman

MOST TRUSTED AUTHORITY ON ENTITY FRAMEWORK

@julielerman thedatafarm.com

