Mapping Relationships in EF Core

Julie Lerman
MOST TRUSTED AUTHORITY ON ENTITY FRAMEWORK
@julielerman thedatafarm.com

Module Overview

Learning more about one-to-many relationships

Understanding many-to-many relationships

Tracking additional data in a join entity

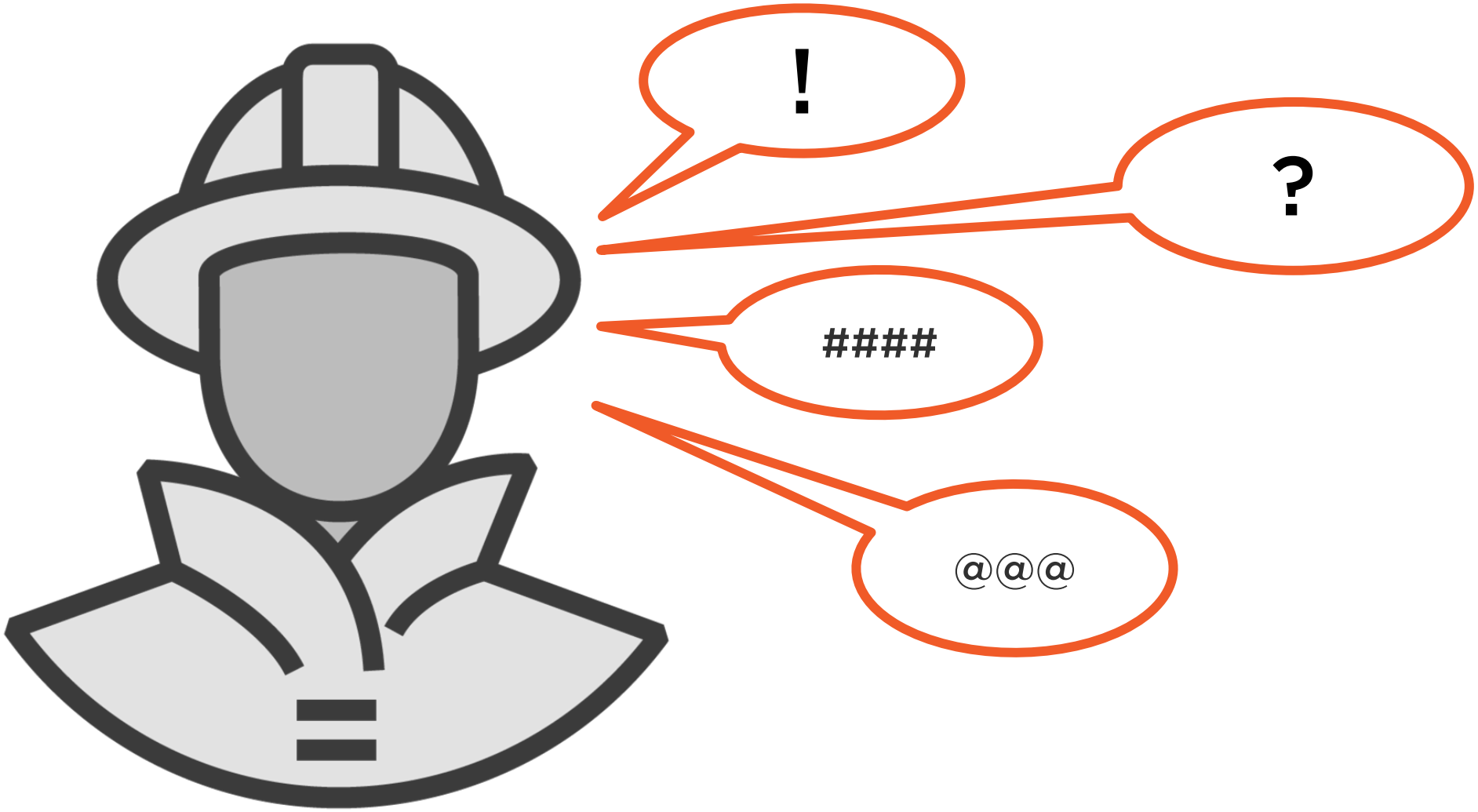
Adding a one-to-one relationship

Visualize the model with EF Core Power Tools

Migrating the database to reflect model changes

Learning More About One-to-Many Relationships

One Samurai Character Can Have Many Quotes



Convention Over Configuration

Default behavior that can be overridden using configurations.

List<Child>
in Parent

Child needs no references back to Parent not required for EF Core to understand 1:* relationship

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

public class Quote
{
  public int Id {get;set;}
  public string Text {get;set;}
}
```

List<Child>
in Parent

Child has a navigation property back to parent. Property is required by default.

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

public class Quote
{
  public int Id {get;set;}
  public string Text {get;set;}
  public Samurai Samurai {get;set;}
}
```

List<Child>
in Parent

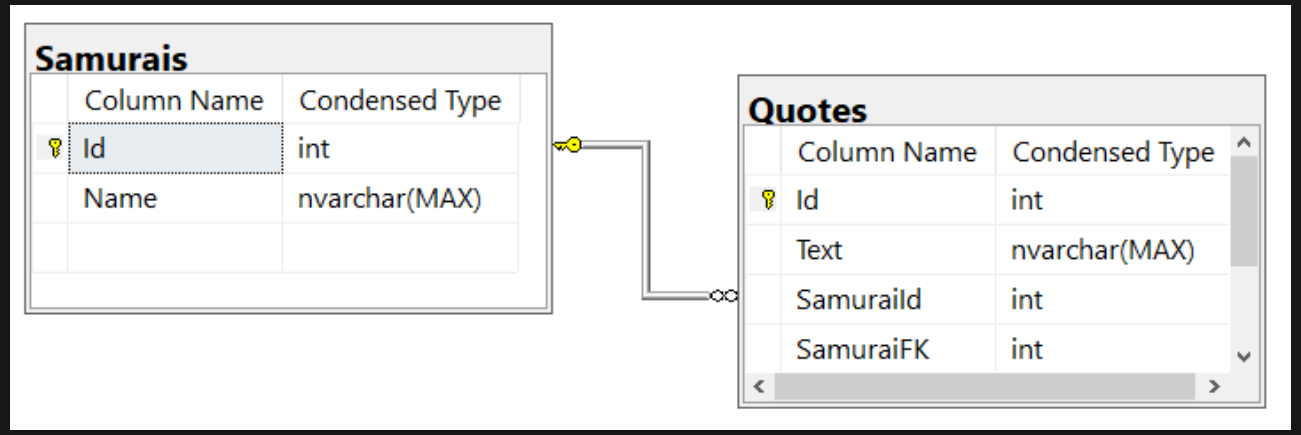
Child has navigation property back to parent and a property recognized as FK

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

public class Quote
{
  public int Id {get;set;}
  public string Text {get;set;}
  public Samurai Samurai {get;set;}
  public int SamuraiId {get;set;}
}
```

```
public class Quote
{
  public int Id {get;set;}
  public string Text {get;set;}
  public Samurai Samurai {get;set;}
  public int SamuraiFK {get;set;}

}
```



Watch Out for Non-Conventional FK Properties

EF Core will see SamuraiFK as a random int column

And EF Core will infer a FK column named SamuraiId

You can override with mappings, to configure SamuraiFK as the foreign key

List<Child>
in Parent

Child's (properly named) FK property with no navigation is recognized as FK

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

public class Quote
{
  public int Id {get;set;}
  public string Text {get;set;}
  public int SamuraiId {get;set;}
}
```

```
public class Quote
{
     public int Id {get;set;}
     public string Text {get;set;}
     public Samurai Samurai {get;set;}
}

public class Quote
{
     public int Id {get;set;}
     public string Text {get;set;}
     public Samurai Samurai {get;set;}
     public int SamuraiId {get;set;}
}

public class Quote
{
     public int Id {get;set;}
     public string Text {get;set;}
     public int SamuraiId {get;set;}
}
```

- Only a navigation property to Samurai
  You must have a samurai object to connect a quote

```
samurai.Quotes.Add(thequote)
thequote.Samurai=someobject
```

- With a foreign key property you don't need a samurai object

```
thequote.SamuraiId=1
```

- And you can even eliminate the navigation property if your logic doesn't need it

Will I ever retrieve a Quote and want to know who said it?

Quote.Samurai.Name

A Few Configuration Examples

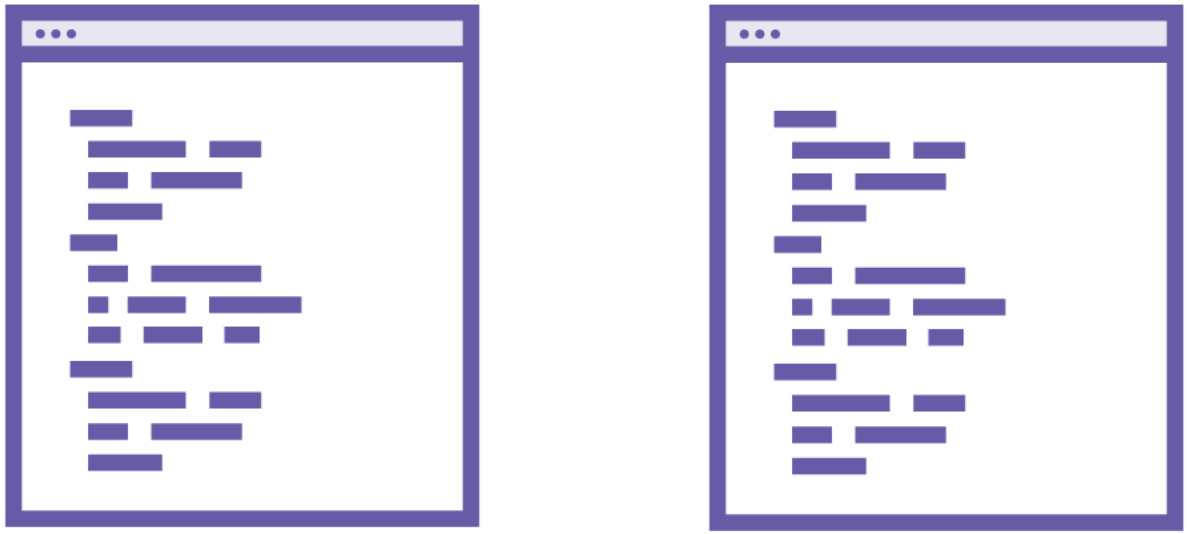
```
modelBuilder.Entity<Samurai>()
.HasMany(s => s.Quotes)
.WithOne(q => q.Samurai)
.HasForeignKey(q => q.SamuraiFK);
```

```
modelBuilder.Entity<Samurai>()
.HasMany(s => s.Quotes)
.WithOne(q => q.Samurai)
.HasForeignKey(q => q.SamuraiFK);
```

EF Core's Default Many-to-Many Mapping

Many-to-Many support has changed dramatically in EF Core 5

Many-to-Many with Skip Navigations

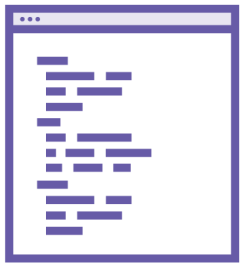


Samurai **Battle**

List<Battle> ⟷ List<Samurai>

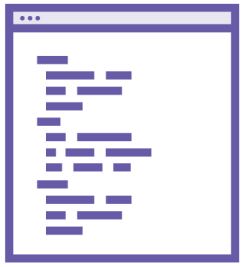
EF Core 5 Can Interpret Skip Navigations

Join Ends with Class Properties



Samurai

List<Battle>



Battle

List<Samurai>

Relational Database: Join Table

Samurais

SamuraiId

BattleSamurai

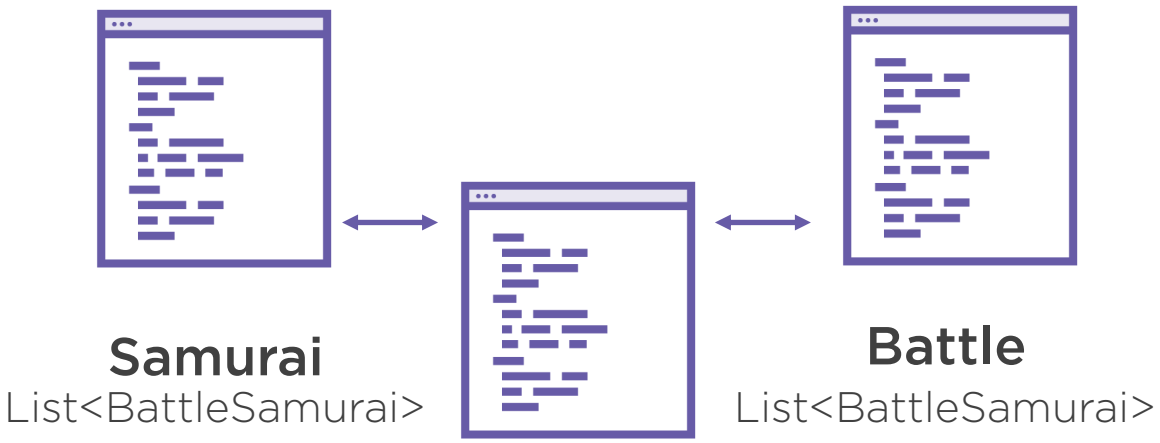
SamuraiId
BattleId

Battles

BattleId

Many-to-Many with Join Entities

Explicit Join Entity



Samurai
List<BattleSamurai>

BattleSamurai
SamuraiId
BattleId

Battle
List<BattleSamurai>

Relational Database: Join Table

Samurais
SamuraiId

BattleSamurai
SamuraiId
BattleId

Battles
BattleId

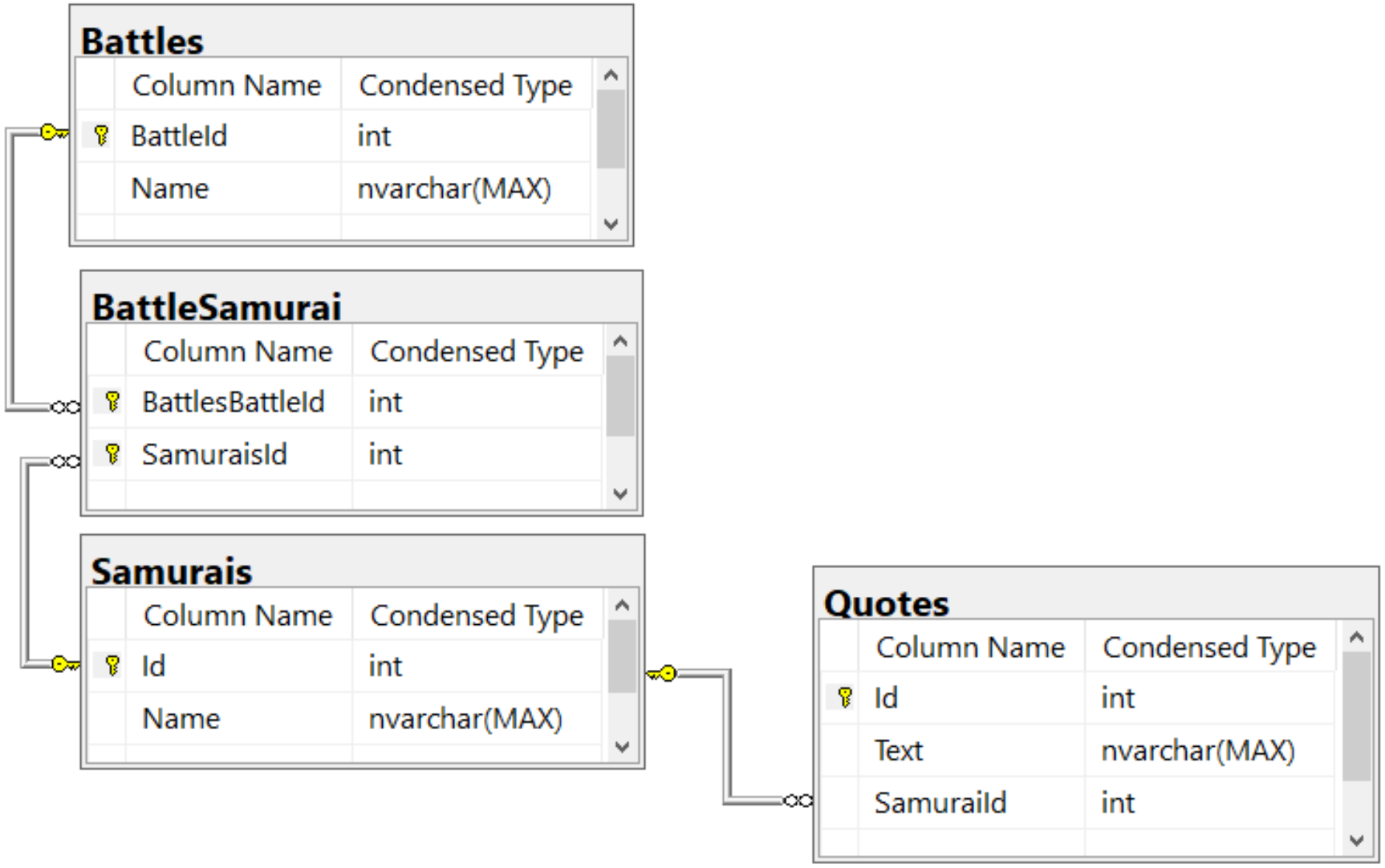
The way EF Core 5 handles many-to-many is much smarter and more flexible than EF6

EF Core 5's New Many-to-Many Support

Much simpler than EF Core 1-3

Smarter & more flexible than EF6

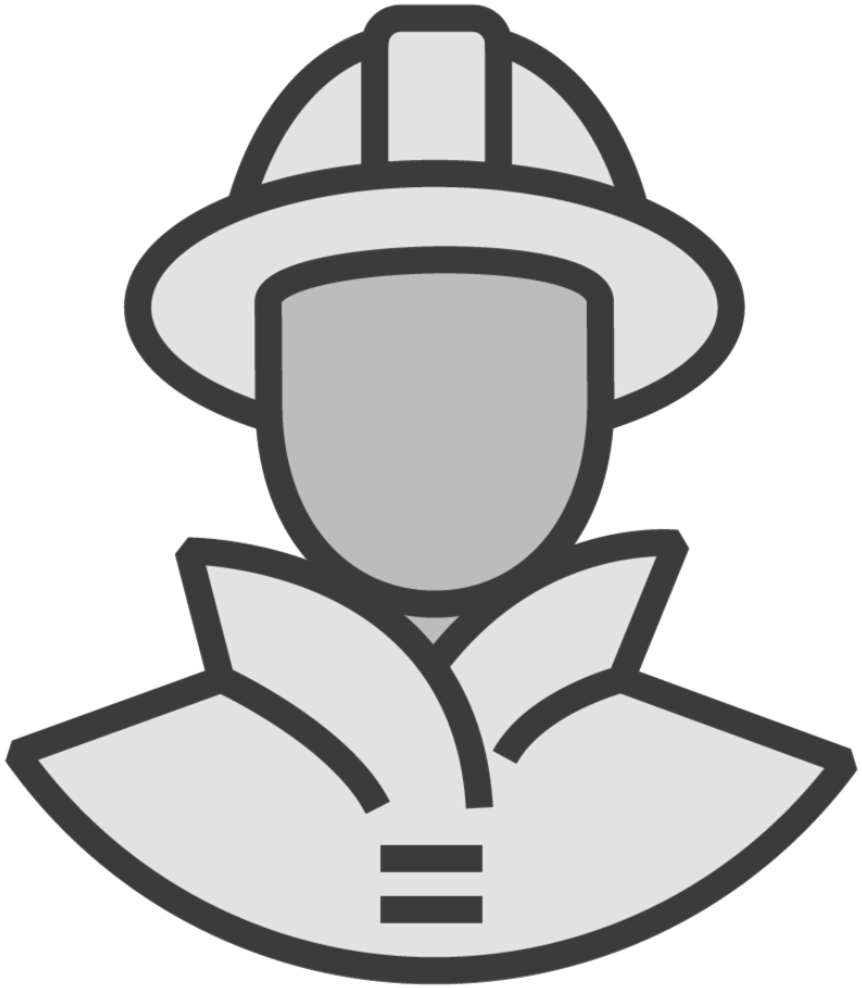
SSMS Diagram of SamuraiAppData Tables



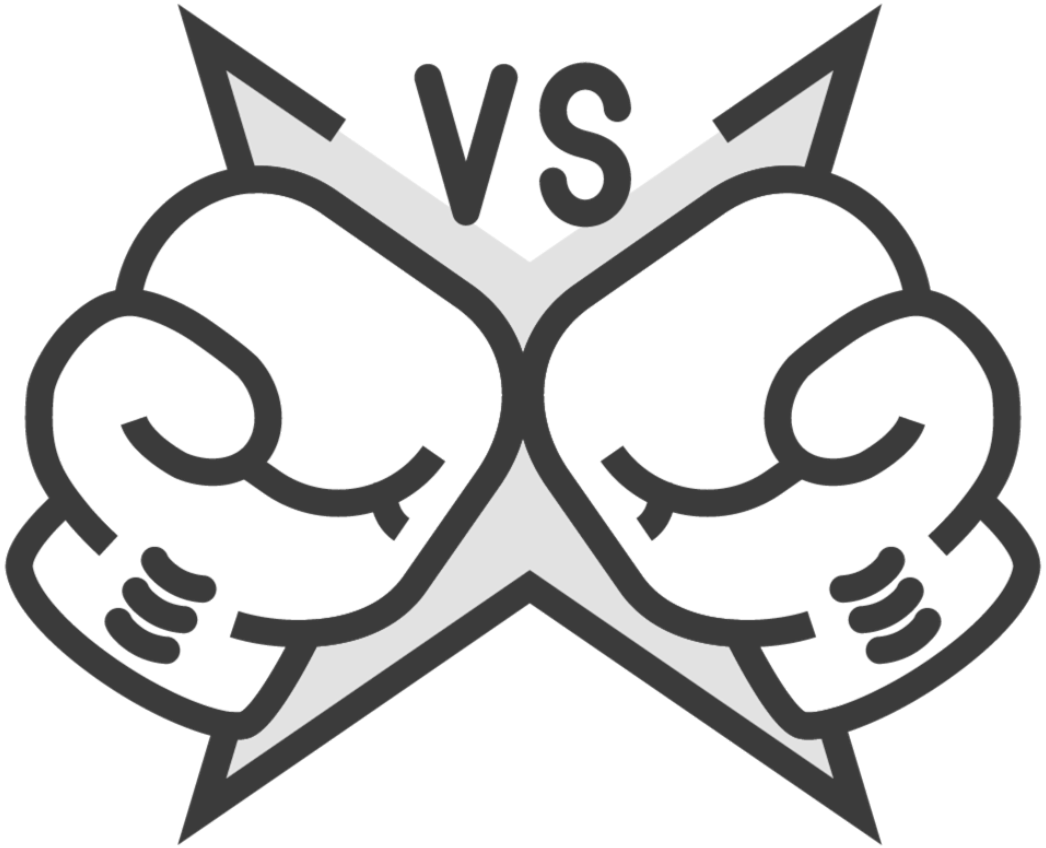
Storing Additional Data with Many-to-Many Payloads

This is a little more advanced, but a common and important capability.

Adding More Info to the Relationship



Samurai Kikuchiyo

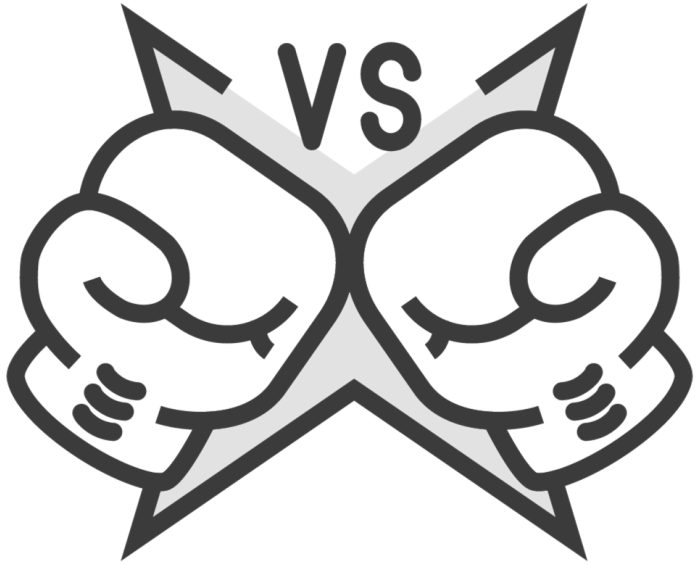


Battle of Anegawa

Adding More Info to the Relationship

Samurai Kikuchiyo

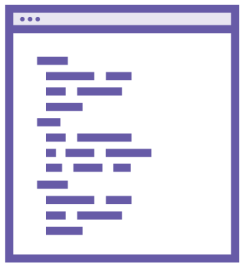
When?



Battle of Anegawa

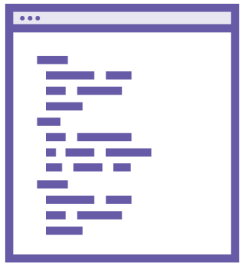
Many-to-Many Relationship

Join Ends with Class Properties



Samurai

List<Battle>



Battle

List<Samurai>

Relational Database: Join Table

Samurais

SamuraiId

BattleSamurai

SamuraiId
BattleId

Battles

BattleId

Many-to-Many Relationship with Payload

Explicit Join Table with Payload

Samurai

List<Battle>

BattleSamurai

SamuraiId
BattleId
JoinDate

Battle

List<Samurai>

Relational Database: Join Table

Samurais

SamuraiId

BattleSamurai

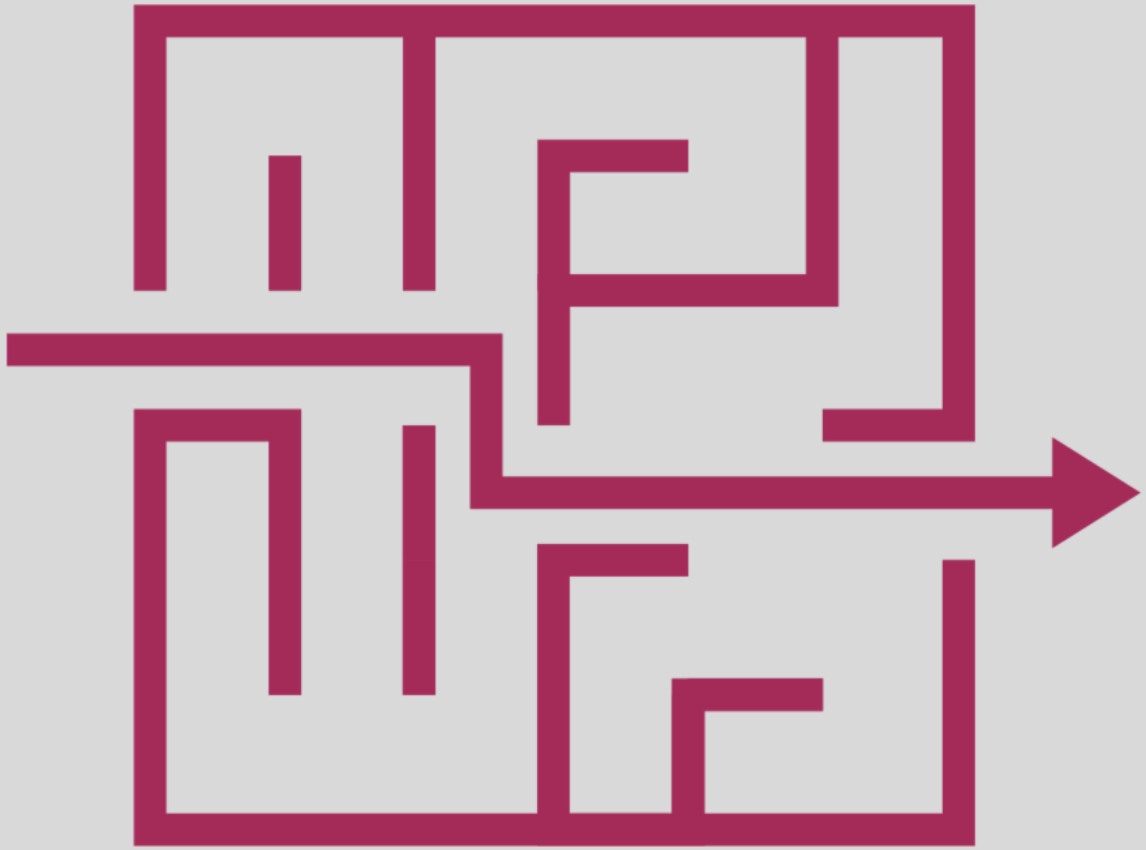
SamuraiId
BattleId
JoinDate

Battles

BattleId

With EF Core 5, the addition of a join entity will not complicate EF's understanding of your relationship

The Path of Least Resistance



EF Core 5: Building on the Foundation

By **Julie Lerman**

Hot on the tail of EF Core 3 is EF Core 5. Remember when we skipped from EF v1 to EF4 to align with .NET Framework 4.0? Like that time, this new gap in version numbers was designed to align with the successor to .NET Core 3. Except that successor is no longer called .NET Core, it's .NET 5. Yet EF Core 5 is keeping the "Core" (as is ASP.NET Core 5). This allows us to continue differentiating from the pre-Core versions of EF and ASP.NET. So, EF Core 5 it is. Happily for me, this article is about EF Core 5 and I'm not charged with further explaining the name and other changes to .NET 5, details you can read about in another article in this issue.

When EF Core 5 was still a few months from release, Arthur Vickers (from the EF team) tweeted some impressive stats about the 637 GitHub issues already closed for the EF Core 5 release:

- 299 bugs fixed
- 113 docs and cleanup
- 225 features and minor enhancements

Building on the EF Core 3 Foundation

You may have read my article introducing CODE Magazine readers to EF Core 3, entitled "Entity Framework Core 3.0, A Foundation for the Future" (https://codemag.com/Article/1911062/Entity-Framework-Core-3.0-A-Foundation-for-the-Future). Because the first iteration of EF Core was a complete rewrite of Entity Framework, it had been a very "v1" version. EF Core 2 goals were around tightening down the existing code base and adding critical features, making EF Core truly production ready. With that stable version in place (and widely used), the team felt that with EF Core 3, it was safe to make the kind of low-level changes that would result in breaking changes. There weren't many new features in EF Core 3, so teams could continue to use EF Core 2 if they wanted while EF Core 3 prepared the framework for a

Julie Lerman

Julie Lerman is a Microsoft Regional director, Docker Captain, and a long-time Microsoft MVP

CODE Magazine

Code Focus Nov 2020

codemag.com/Article/2010042

Source: #wocintechchat
flickr.com/photos/wocintechchat

Many-to-Many Relationship with Payload

Explicit Join Table with Payload

Samurai

List<Battle>

BattleSamurai

SamuraiId
BattleId
JoinDate

Battle

List<Samurai>

Relational Database: Join Table

Samurais

SamuraiId

BattleSamurai

SamuraiId
BattleId
JoinDate

Battles

BattleId

Many-to-Many Relationship with Payload

You must configure the DbContext to understand this mapping

Explicit Join Table with Payload

Samurai BattleSamurai Battle

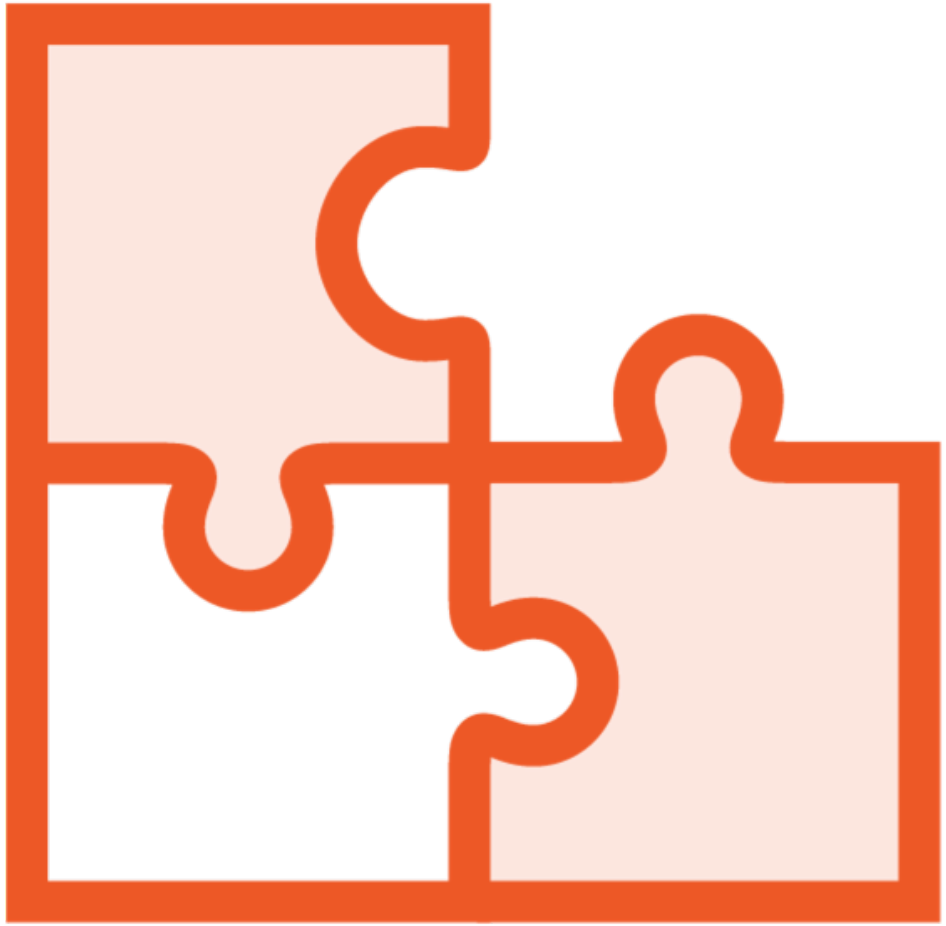
Relational Database: Join Table

Samurais BattleSamurai Battles

Configuring the Many-to-Many Payload

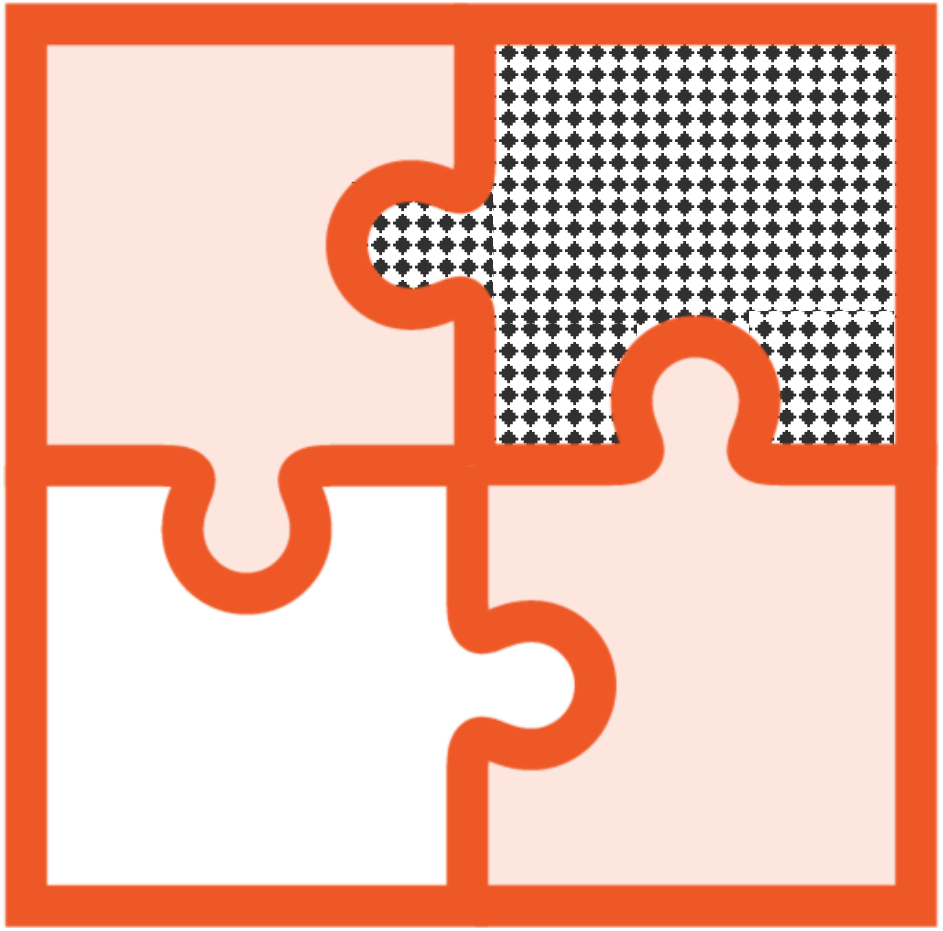
Configuring the Join Entity

Identify the known many-to-many relationship



Configuring the Join Entity

Identify the known many-to-many relationship



Add more information to that mapping

```
modelBuilder
  .Entity<End1>
  .HasMany(e1=>e1.E2List )
  .WithMany(e2=>e2.E1List)
```

Express *:* Using HasMany/WithMany

End #1 HAS MANY End #2s via the list navigation property.

That other end also is WITH MANY of End1, represented by the E1 list property

Start with either end. There is no difference.

Migrating the Many-to-Many Payload

HasColumnName Fluent Mapping

```
modelBuilder.Entity<BattleSamurai>()
     .Property(bs => bs.SamuraiId).HasColumnName("SamuraisId");

modelBuilder.Entity<BattleSamurai>()
     .Property(bs => bs.BattleId).HasColumnName("BattlesBattleId");
```

```
public DbSet<Battle> Battle
 {get;set;}


public class BattleSamurai
{
}


modelBuilder
.Entity<BattleParticipant>()
.ToTable("BattleSamurai")
```

Table Mapping Conventions

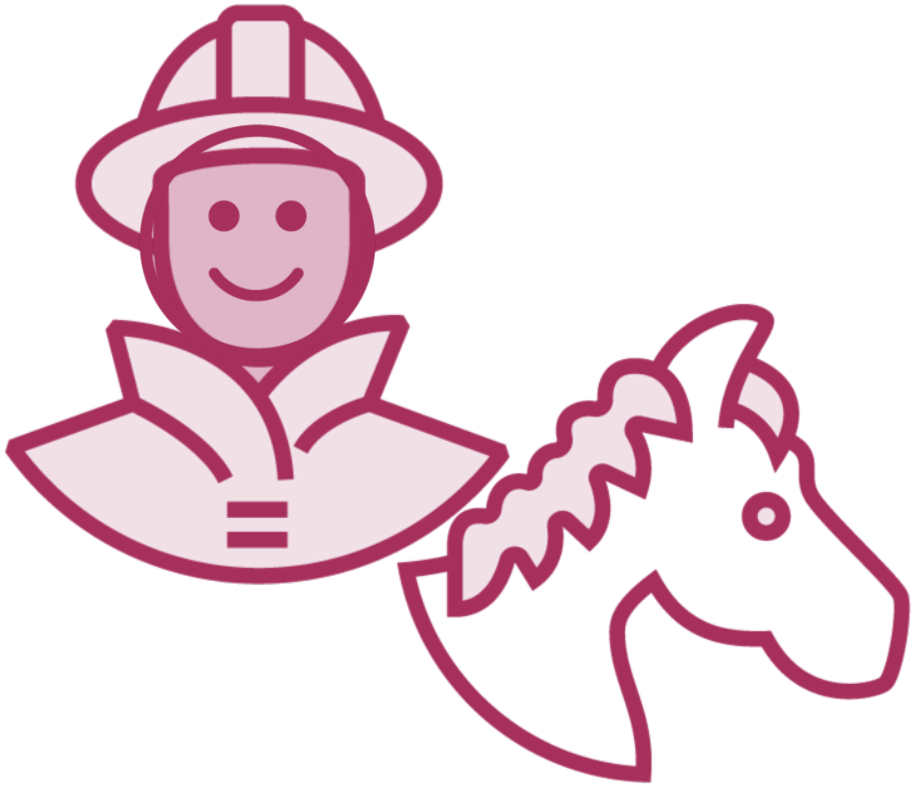
- DbSet drives table name (Samurais)
- If there is no DbSet, then table uses the name of the class (BattleSamurai)
- Override with ToTable mapping

Adding a One-to-One Relationship

By Thierry Bernard [CC BY-SA 3.0 (https://creativecommons.org/licenses/by-sa/3.0)], via Wikimedia Commons

Introducing a new class: Horse

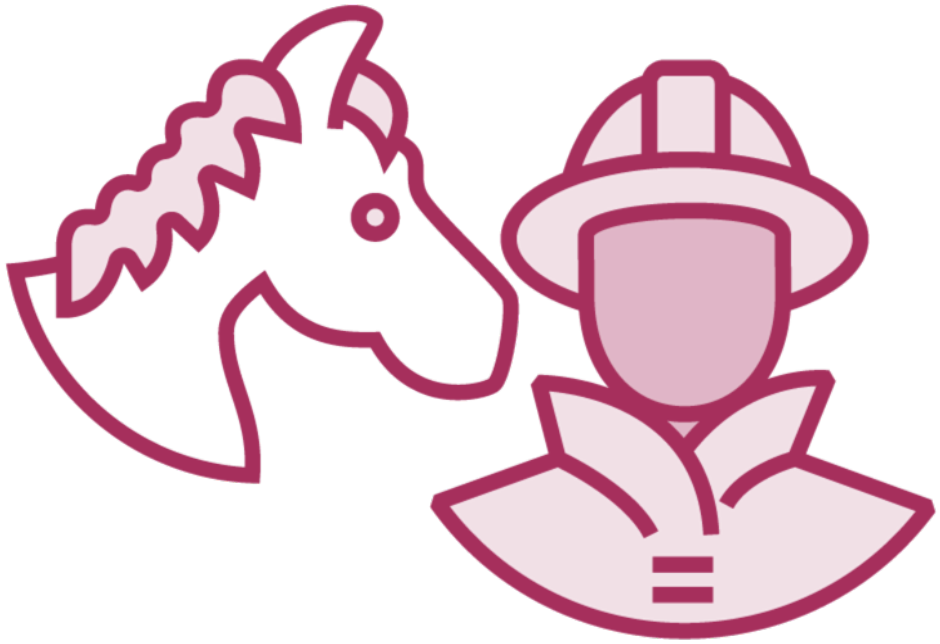
Dependent End of 1:1 is Always Optional



Samurai can have a horse

Samurai can be without a horse

Dependent Default: Must Have a Parent



Horse must have a Samurai

Additional Considerations

Integer FK is non-nullable by default

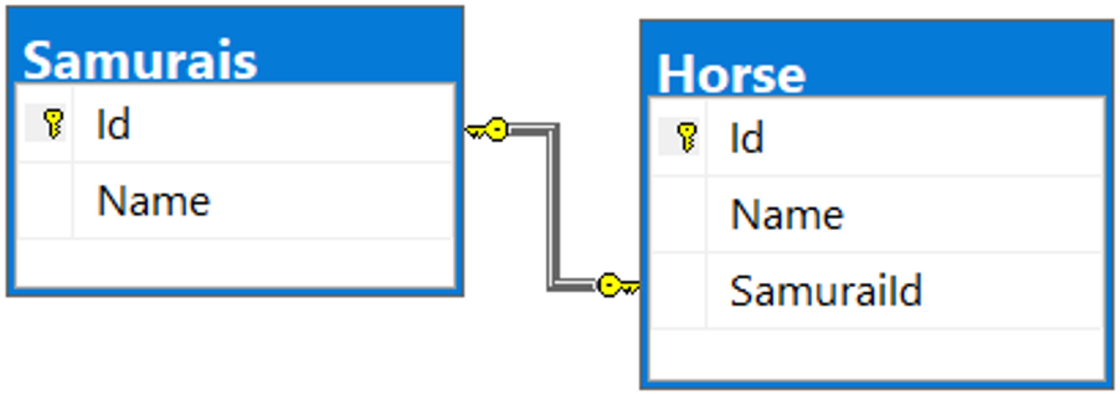
Allow “orphaned” horses with nullable foreign key or a mapping

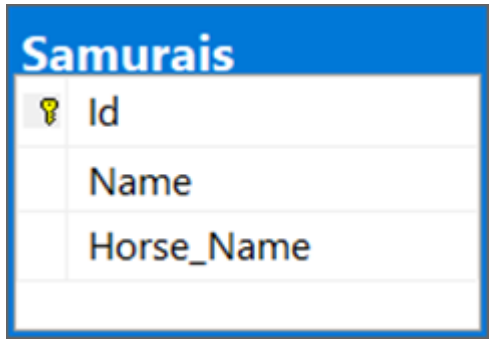
EF Core can usually determine principal & dependent

Fluent API mappings can be used to adjust EF Core’s conventional interpretation

Visualizing How EF Core Sees Your Model

One-to-One Table Mappings





Default

Dependent gets its own table

Advanced

Dependent data in principal table

Review

- Learned more about 1:* relationships
- Learned about EF Core 5's conventional many-to-many support
- Implemented a *:* with payload data
- Learned about 1:1 defaults e.g., optional dependents and separate tables
- Visualized model with EF Core Power Tools
- More tricks with Fluent API and migrations
- Migrated database to apply model changes

Coming Up

Interacting with your EF Core data model and logging

Resources

Entity Framework Core on GitHub github.com/dotnet/efcore

EF Core Documentation docs.microsoft.com/ef

EF Core Power Tools on GitHub github.com/ErikEJ/EFCorePowerTools/wiki

Entity Framework Community Standup - Many-to-Many in EF Core 5.0
youtube.com/watch?v=W1sxepfIMRM

Pluralsight EF Core 2: Mappings (includes Explicit Join Entity Mapping) bit.ly/2LppcMj

Mapping Many-to-Many and One-to-One Relationships

Julie Lerman
MOST TRUSTED AUTHORITY ON ENTITY FRAMEWORK
@julielerman thedatafarm.com