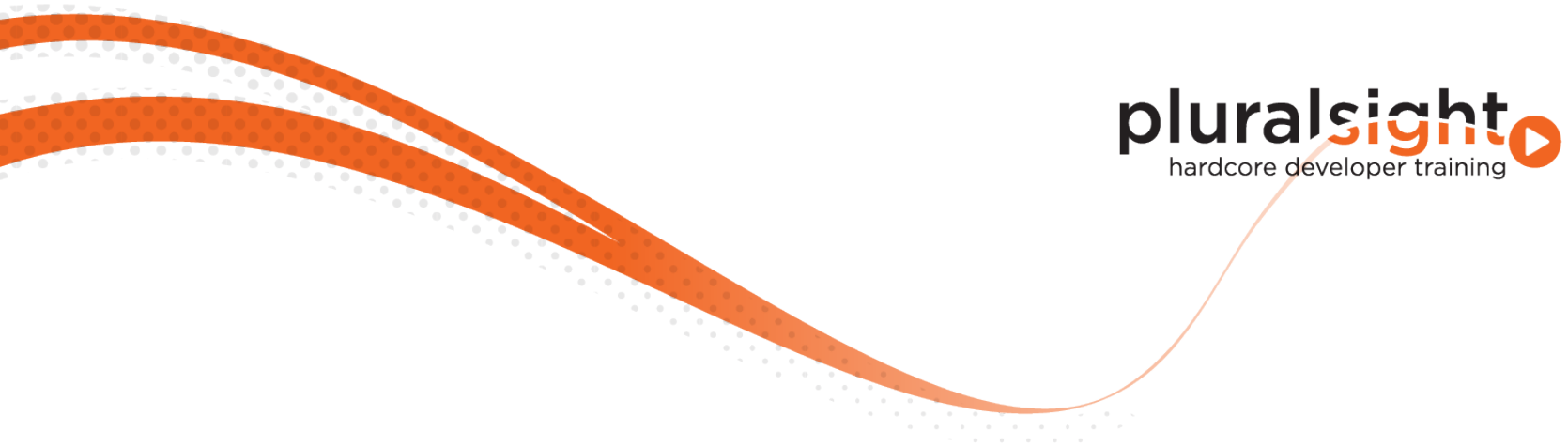
Data Models, Entity Framework, and Data Patterns

John Papa



pluralsight
hardcore developer training

Agenda

- Data Layer Technologies
- Creating Models
- Creating the Entity Framework DbContext
- Creating Repositories
- Creating a Unit of Work

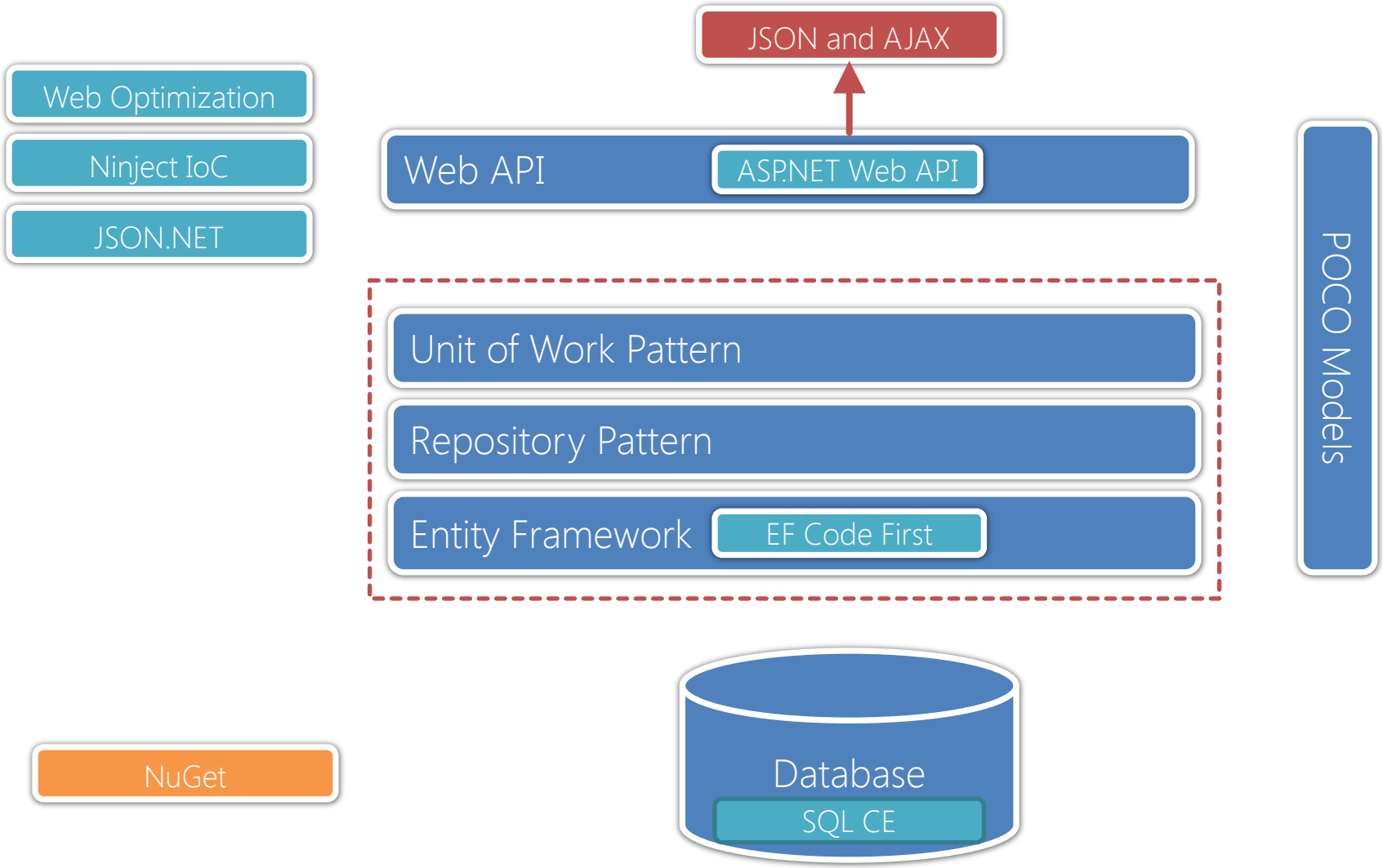
Agenda

- Data Layer Technologies
- Creating Models
- Creating the Entity Framework DbContext
- Creating Repositories
- Creating a Unit of Work

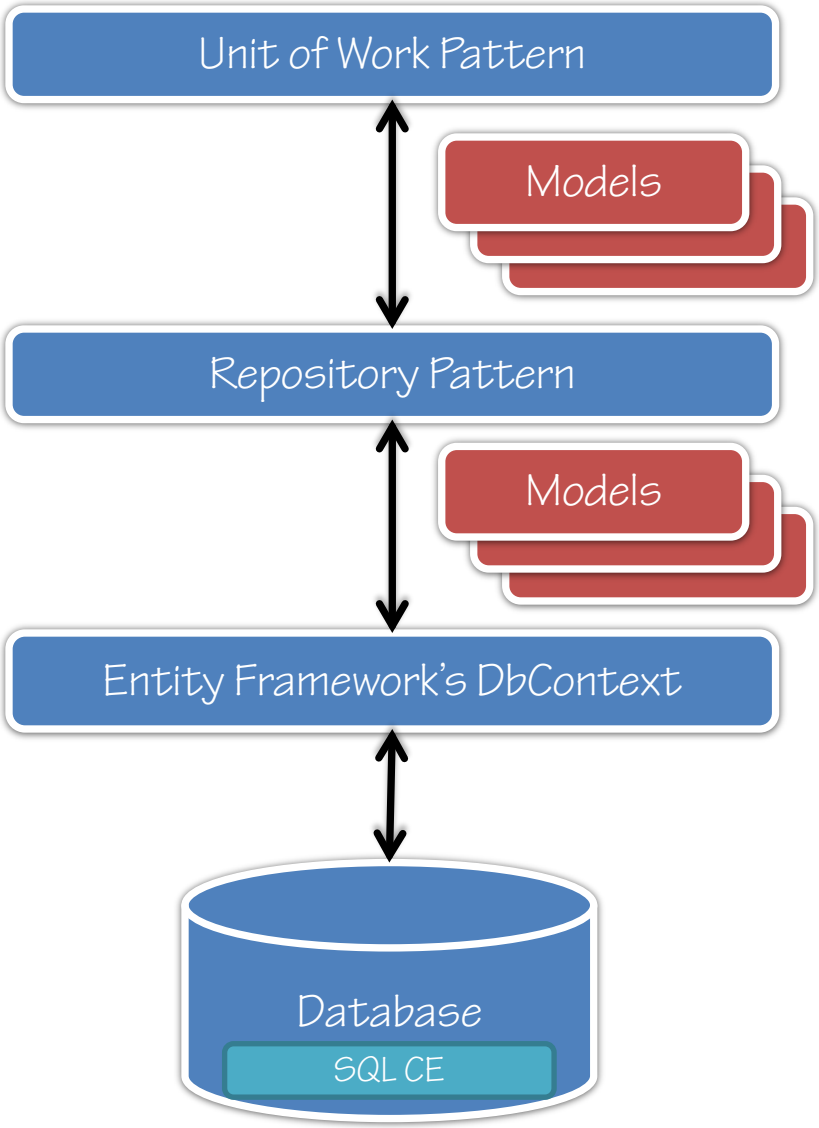
Data Technologies

- **Data stored in SQL Server**
- **Object Relational Mapper (ORM)**
  - Entity Framework Code First
  - Data is stored and saved to its own context (DbContext)
- **Repositories expose the ORM's data for retrieval/saving to the context**
- **Unit of Work aggregates the repositories, so we can commit and cancel multiple changes to repositories**

Server Technologies



Data Layer Patterns

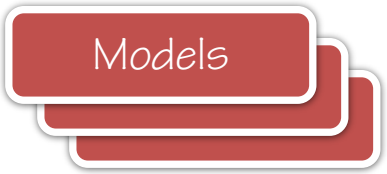


Agenda

- Data Layer Technologies
- Creating Models
- Creating the Entity Framework DbContext
- Creating Repositories
- Creating a Unit of Work

What's a Model?

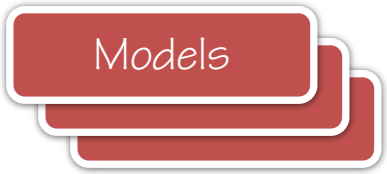
- **Data containers / stores**
  - Properties that define your data
  - Vehicle for your data



- **Entity Framework Fills the Vehicles**

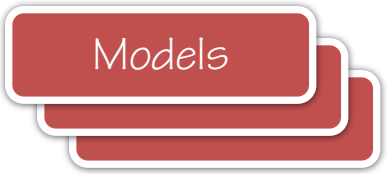
POCO

- **Plain Old Class (CLR) Object**
- **Independent and stand alone objects**
- **Makes them easy to pass around**



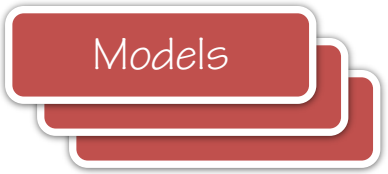
Web API | Unit of Work | Repositories | Entity Framework

Defining a Model



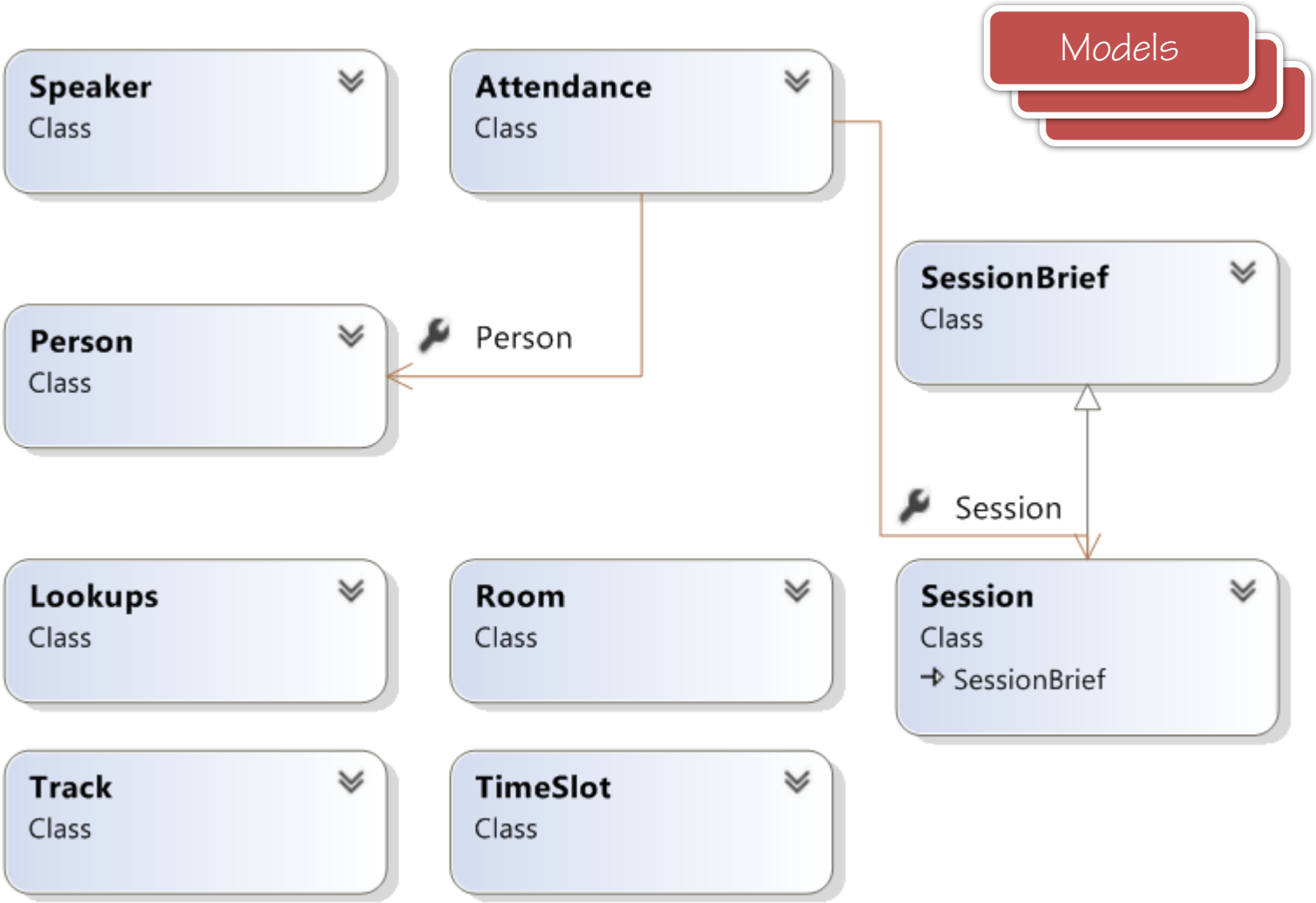

```
public class SessionBrief
{
        public int Id { get; set; }
        public string Title { get; set; }
        public string Code { get; set; }
        public int SpeakerId { get; set; }
        public int TrackId { get; set; }
        public int TimeSlotId { get; set; }
        public int RoomId { get; set; }
        public string Level { get; set; }
        public string Tags { get; set; }
}
```

Defining a Model - Inheritence




```
public class Session : SessionBrief
{
        public string Description { get; set; }
        public virtual Person Speaker { get; set; }
        public virtual Track Track { get; set; }
        public virtual TimeSlot TimeSlot { get; set; }
        public virtual Room Room { get; set; }

        public virtual ICollection<Attendance>
            AttendanceList { get; set; }
}
```

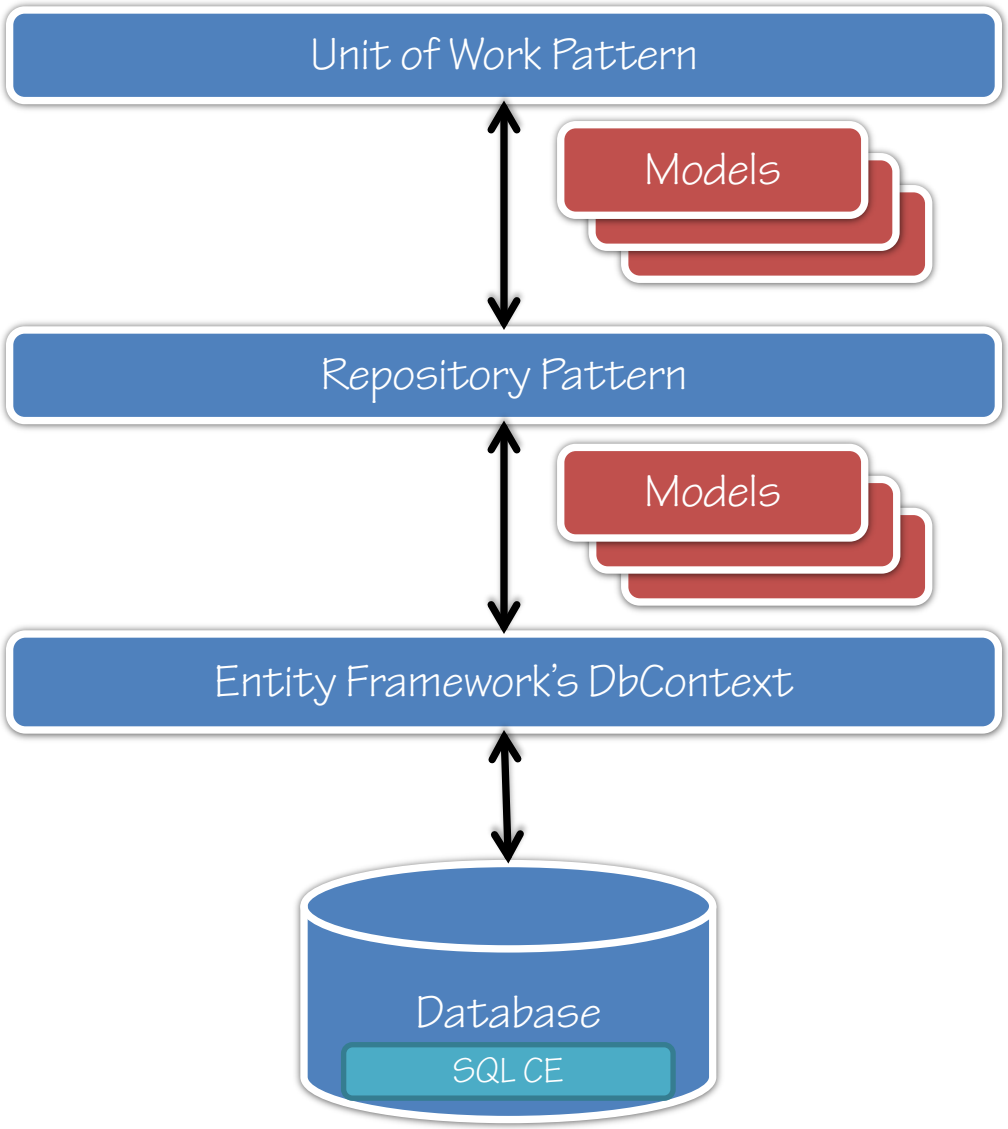
Model Diagram



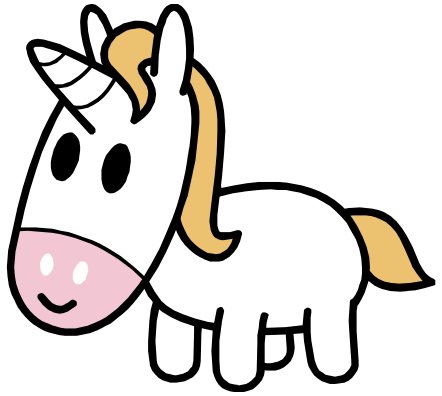
Agenda

- Data Layer Technologies
- Creating Models
- Creating the Entity Framework DbContext
- Creating Repositories
- Creating a Unit of Work

DbContext

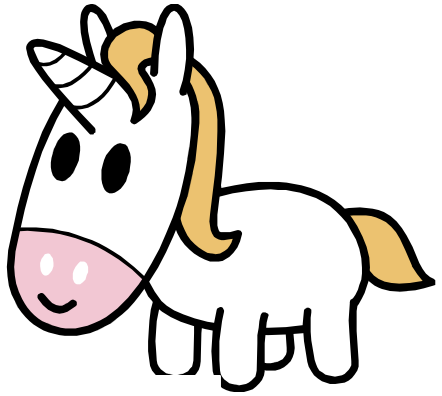

Unit of Work Pattern
Models
Repository Pattern
Models
Entity Framework's DbContext
Database
SQL CE

Roles of the DbContext Class



- **Microsoft ADO.NET Entity Framework Code First**
  - (Magical Unicorn Edition)
- **Defines the ~~magic~~ relations between Models and the Database**
- **Stores objects and changes in its context (in memory)**

Define Sets of Data



- DbSet<T>
  - Defines the relation between tables and models

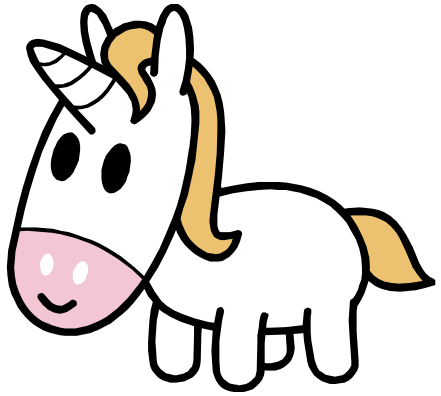
```
public DbSet<Session> Sessions { get; set; }

public DbSet<Person> Persons { get; set; }
public DbSet<Attendance> Attendance { get; set; }
public DbSet<Room> Rooms { get; set; }
public DbSet<TimeSlot> TimeSlots { get; set; }
public DbSet<Track> Tracks { get; set; }
```

Configuring the DbContext

- **Configure entity relations**
  - ex: 1 – many
  - Mappings between tables or fields

```
public class SessionConfiguration
    : EntityTypeConfiguration<Session>
{
    public SessionConfiguration()
    {
        HasRequired(s => s.Speaker)
            .WithMany(p => p.SpeakerSessions)
            .HasForeignKey(s => s.SpeakerId);
    }
}
```

Defining Conventions with the DbContext



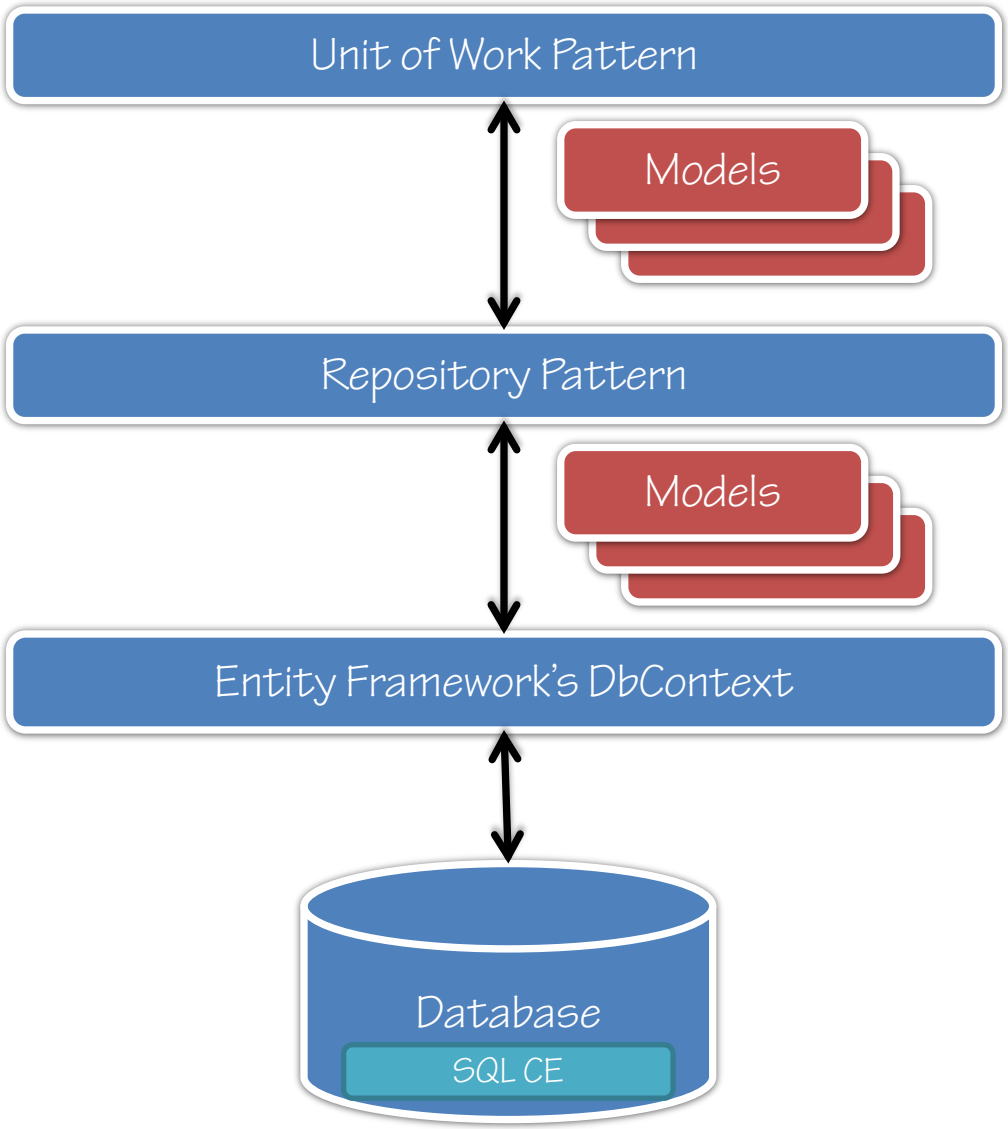
- **Establish Seed Data**
  - CodeCamper will generate the database if it doesn't exist
- **Define conventions**
  - Plural or singular table names

```
// Use singular table names
modelBuilder.Conventions
    .Remove<PluralizingTableNameConvention>();
```

Agenda

- Data Layer Technologies
- Creating Models
- Creating the Entity Framework DbContext
- Creating Repositories
- Creating a Unit of Work

Repositories



Why Use a Repository?

- **Maintenance**
  - Data access code is easy to find, debug, change
- **Code Reuse**
- **Focused on getting and saving data**
- **Consistent API**
- **Single Responsibility Principle (SRP)**

Repository to Simplify the API

- Expose DbContext features for CRUD
- Consistent API
- Simplifies the way calls are made to the DbContext

Consistency

```
public interface IRepository<T> where T : class
{
    IQueryable<T> GetAll();
    T GetById(int id);
    void Add(T entity);
    void Update(T entity);
    void Delete(T entity);
    void Delete(int id);
}
```

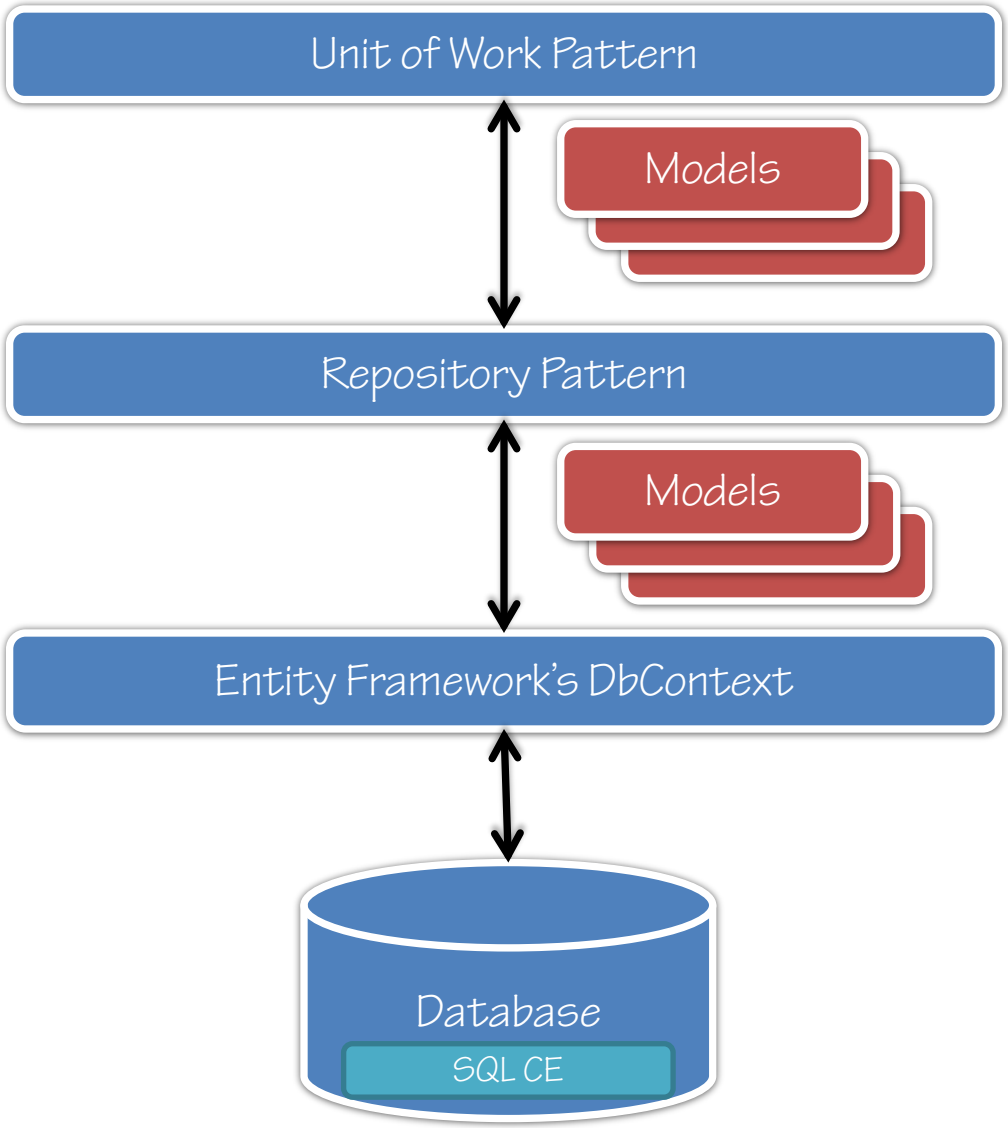
Repository Base Class

```
public class EFRepository<T> : IRepository<T> where T : class
{
    protected DbContext DbContext { get; set; }
    protected DbSet<T> DbSet { get; set; }

    public virtual T GetById(int id)
    {
        return DbSet.Find(id);
    }
    // Code shortened for slides
}
```

Agenda

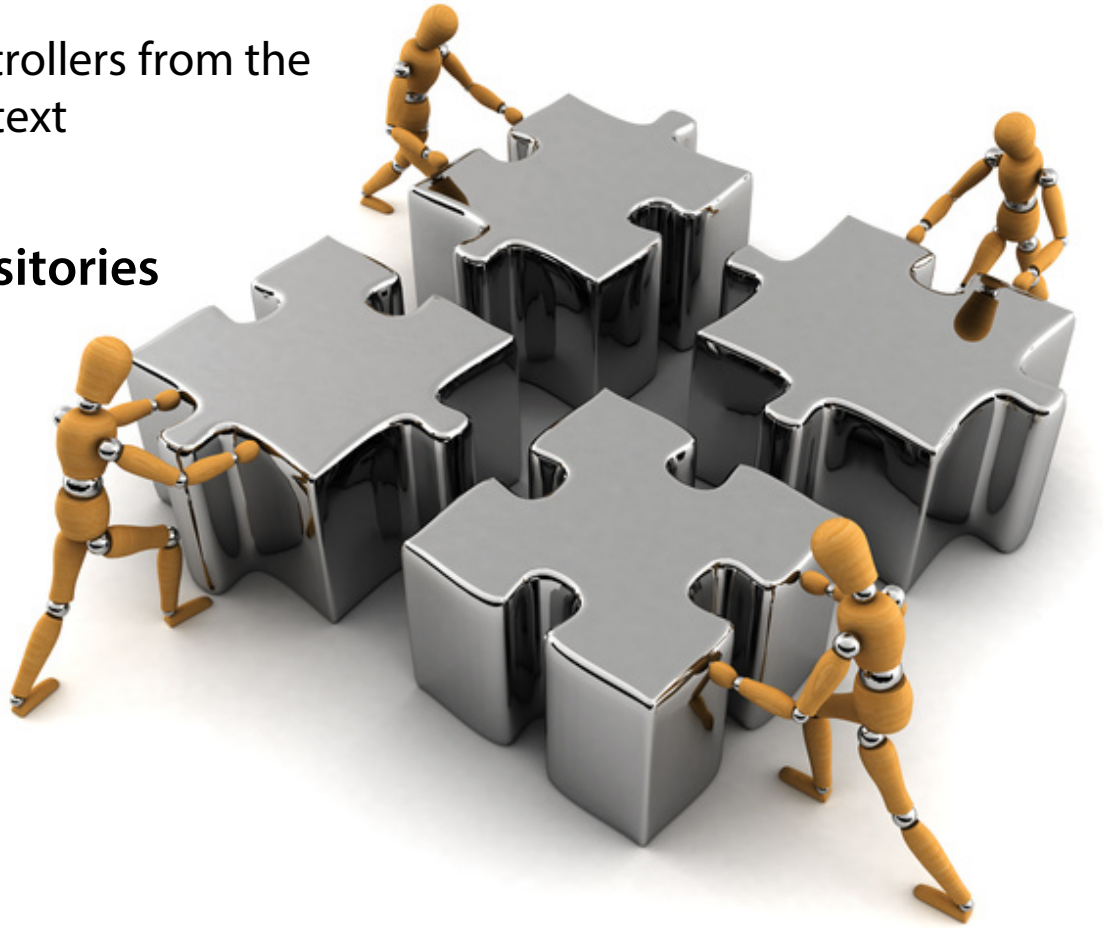
- Data Layer Technologies
- Creating Models
- Creating the Entity Framework DbContext
- Creating Repositories
- Creating a Unit of Work

Unit of Work


Unit of Work Pattern
Models
Repository Pattern
Models
Entity Framework's DbContext
Database
SQL CE

Unit of Work Pattern

- **Puts it all together**
  - Decouples Web API Controllers from the Repositories and DbContext
- **Aggregates calls to Repositories**
- **Simple interface**
  - Series of IRepository<T>
  - Commit



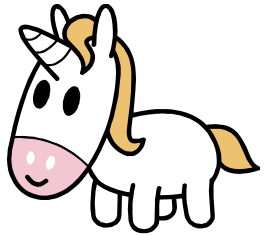
Unit of Work Class

```
public class CodeCamperUow : ICodeCamperUow, IDisposable
{
    public IRepository<Room> Rooms
        { get { return GetStandardRepo<Room>(); } }
    public ISessionsRepository Sessions
        { get { return GetRepo<ISessionsRepository>(); } }

    // Code shortened for slides

    public void Commit()
    {
        DbContext.SaveChanges();
    }
```

Summary

- Models are the vehicles for the data

- Entity Framework Code First ‘s DbContext manages the data



- Repositories simplify the operations over the DbContext

- Unit of Work aggregates Repositories and separates Web API from DbContext