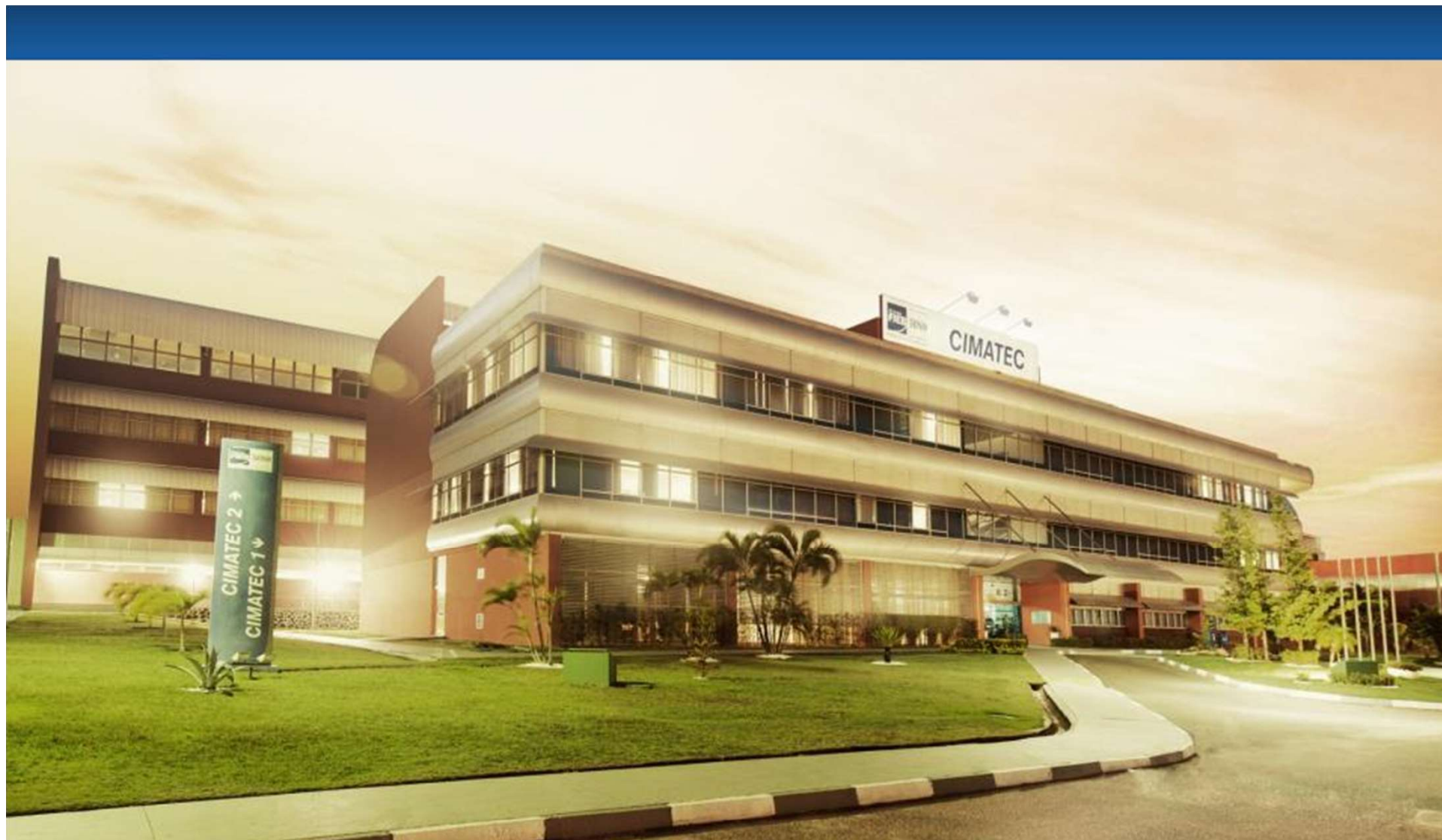


**TECNOLOGIA E INOVAÇÃO
EM PROL DA INDÚSTRIA**



Curso Técnico em Informática

Desenvolvimento de Sistemas II – 180h

Prof^a: Francisleide Almeida

Data annotations

- Classe :
System.ComponentModel.DataAnnotations
- Os atributos usados passarão por validação client-side;
- Especificará informações “extras” no banco de dados.



Data annotations

- Table name
- Key
- Required
- MaxLength
- MinLength
- StringLength

`System.ComponentModel.DataAnnotations`

- Column
- ForeignKey

`System.ComponentModel.DataAnnotations.Schema`

Table name

Table Attribute: `[Table(string name, Properties:[Schema = string])]`

- name: Nome da tabela no banco.
- Schema: Nome do banco que a tabela foi criada (opcional).

```
using System.ComponentModel.DataAnnotations.Schema;

[Table("StudentMaster")]
public class Student
{
    public int StudentID { get; set; }
    public string StudentName { get; set; }
}
```

Key

- O atributo Key pode ser aplicado a uma propriedade de uma classe para torná-la uma chave primária;
- Se for criado atributo com a palavra ID ou nome da classe seguido de ID é entendido como chave primária auto-increment.

```
using System.ComponentModel.DataAnnotations;

public class Student
{
    [Key]
    public int StudentKey { get; set; }
    public string StudentName { get; set; }
}
```

Key

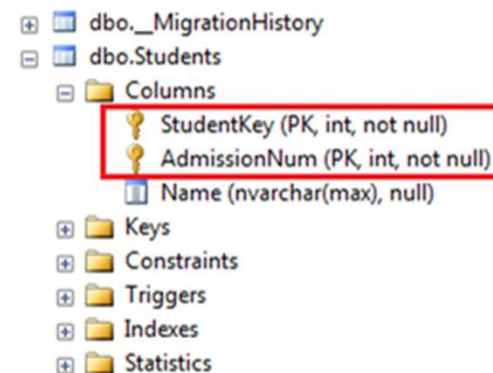
Para criar chave primária composta:

```
using System.ComponentModel.DataAnnotations;

public class Student
{
    [Key]
    [Column(Order=1)]
    public int StudentKey { get; set; }

    [Key]
    [Column(Order=2)]
    public int AdmissionNum { get; set; }

    public string StudentName { get; set; }
}
```



Required

Campo obrigatório no banco - NOT NULL

```
01. [Required]
02. public string FirstName
03. {
04.     get;
05.     set;
06. }
07. [Required]
08. public string LastName
09. {
10.     get;
11.     set;
12. }
```

Para permitir campo nulo, insira uma interrogação após o tipo do atributo:

```
public int? idade { get; set; }
```

MaxLength

Especifica o valor máximo permitido para o campo

```
using System.ComponentModel.DataAnnotations;

public class Student
{
    public int StudentID { get; set; }
    [MaxLength(50)]
    public string StudentName { get; set; }
}
```

MinLength

Especifica o valor mínimo para a string.

StringLength

Adiciona um tamanho máximo para a String
(varchar[60])

```
01. [Required]
02. [StringLength(60)]
03. public string LastName
04. {
05.     get;
06.     set;
07. }
```

FirstName

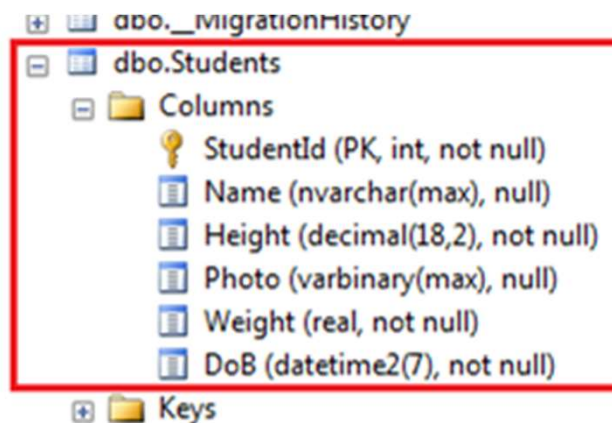
LastName

The field LastName must be a string with a maximum length of 60.

Column

Pode ser aplicado em uma ou mais colunas do banco para configurar o nome, a ordem que ela será criada na tabela e o tipo de dados.

```
[Column (string name, Properties:[Order = int],  
[TypeName = string]
```



```
using System.ComponentModel.DataAnnotations.Schema;  
  
public class Student  
{  
    [Column(Order = 0)]  
    public int StudentID { get; set; }  
  
    [Column("Name", Order = 1)]  
    public string StudentName { get; set; }  
  
    [Column("DoB", Order = 5)]  
    public DateTime DateOfBirth { get; set; }  
    [Column(Order = 3)]  
    public byte[] Photo { get; set; }  
    [Column(Order = 2)]  
    public decimal Height { get; set; }  
    [Column(Order = 4)]  
    public float Weight { get; set; }  
}
```


ForeignKey

Adiciona referência externa para outra tabela.

É feito automaticamente, quando há referência a outra classe.

```
public class Student
{
    public int StudentID { get; set; }
    public string StudentName { get; set; }

    //Foreign key for Standard
    public int StandardId { get; set; }
    public Standard Standard { get; set; }
}

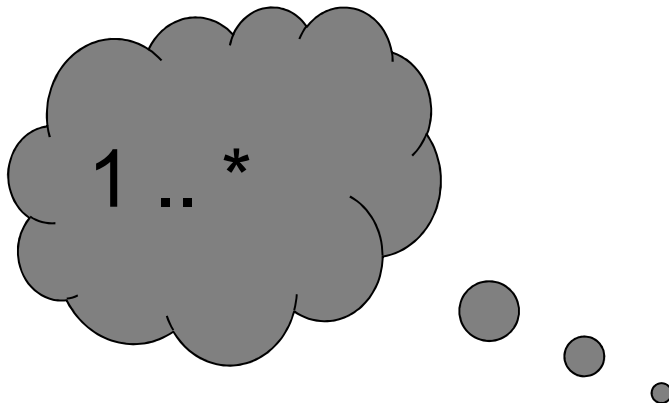
public class Standard
{
    public int StandardId { get; set; }
    public string StandardName { get; set; }

    public ICollection<Student> Students { get; set; }
}
```

ForeignKey

Adiciona referência externa para outra tabela.

É feito automaticamente, quando há referência a outra classe.



```
public class Student
{
    public int StudentID { get; set; }
    public string StudentName { get; set; }

    //Foreign key for Standard
    public int StandardId { get; set; }
    public Standard Standard { get; set; }
}

public class Standard
{
    public int StandardId { get; set; }
    public string StandardName { get; set; }

    public ICollection<Student> Students { get; set; }
}
```

ForeignKey

Criando uma chave estrangeira com o nome do atributo da classe

```
public class Student
{
    public int StudentID { get; set; }
    public string StudentName { get; set; }

    [ForeignKey("Standard")]
    public int StandardRefId { get; set; }
    public Standard Standard { get; set; }
}

public class Standard
{
    public int StandardId { get; set; }
    public string StandardName { get; set; }

    public ICollection<Student> Students { get; set; }
}
```

ForeignKey

Criando chave estrangeira com nome personalizado

```
public class Student
{
    public int StudentID { get; set; }
    public string StudentName { get; set; }

    public int StandardRefId { get; set; }

    [ForeignKey("StandardRefId")]
    public Standard Standard { get; set; }
}

public class Standard
{
    public int StandardId { get; set; }
    public string StandardName { get; set; }

    public ICollection<Student> Students { get; set; }
}
```

Database Migrations

Alterações no banco

Opção 1:

- Apagar o arquivo .mdf do projeto e buildar novamente

Opção 2:

- Utilizar Migration

Alterações no banco

Opção 1:

- Apagar o arquivo .mdf do projeto e buildar novamente

Opção 2:

- Utilizar Migration



Migrations

- Com o Code First Migrations, podemos ter versões da base de dados, voltar versões e manter um histórico;
- Cria métodos de update e downgrade com o código necessário para aplicar as mudanças;
- Pode ser feito de duas formas:
 - Migrations;
 - Automatic Migrations;

Normal Migrations

- O caminho normal da migrations consiste em, por linha de comando, criar uma migration dando um nome para ela e depois rodando o comando de update;
- Caso não seja criada, uma Exception será lançada.

```
Add-Migration NomeDaMigration  
Update-database
```

Normal Migrations

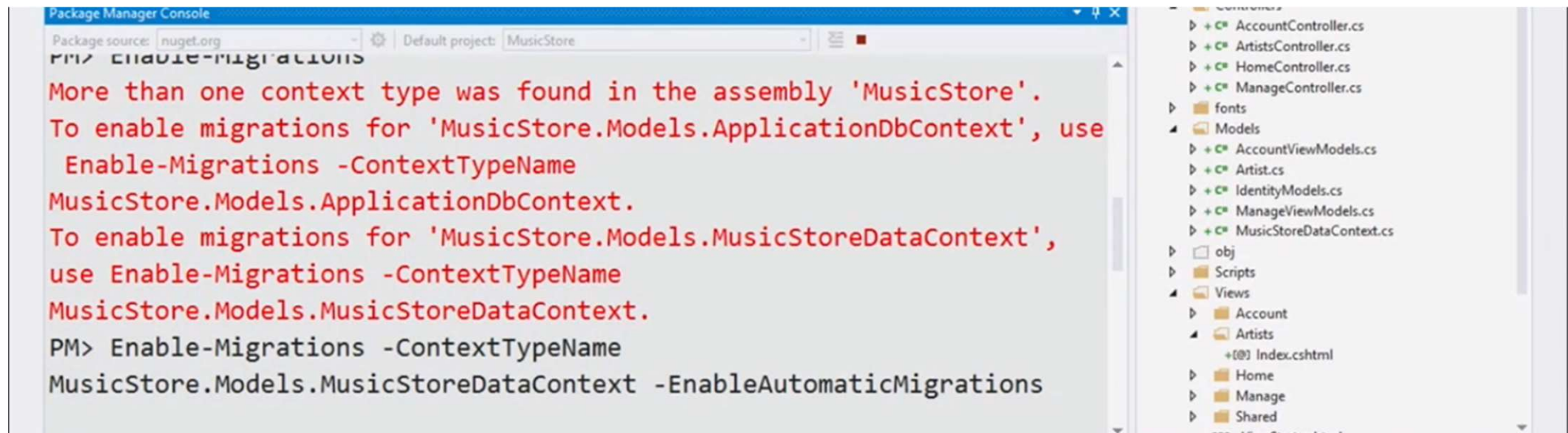
- Ao adicionar uma migration, vai ser adicionado um arquivo com o código da migration na pasta Migrations do seu projeto (o nome é um timestamp seguido do nome que você deu a sua migration)
- É por esse timestamp que a migrations sabe a ordem delas.
- Você pode ver o código SQL gerado usando o parâmetro -Verbose na hora de rodar o update.
- Caso esteja numa versão posterior e queira voltar uma versão, é só, ao rodar o comando update, dizer o nome completo da migration com o parâmetro TargetMigration, incluindo o timestamp

Update-Database -TargetMigration:"201210301049442_second2"

Automatic Migrations

- Ao usar o caminho automatic migrations não é preciso criar uma migration a cada mudança no modelo, somente rodar o comando update.
- Não gera métodos de update e downgrade;

Automatic Migrations



The screenshot displays the Visual Studio interface. On the left, the Package Manager Console window is open, showing the command prompt for the 'MusicStore' project. It contains a warning message about multiple context types and two commands to enable migrations. On the right, the Solution Explorer shows the project structure, including controllers, models, and views.

```
Package Manager Console
Package source: nuget.org | Default project: MusicStore
PM> Enable-Migrations
More than one context type was found in the assembly 'MusicStore'.
To enable migrations for 'MusicStore.Models.ApplicationDbContext', use
Enable-Migrations -ContextTypeName
MusicStore.Models.ApplicationDbContext.
To enable migrations for 'MusicStore.Models.MusicStoreDataContext',
use Enable-Migrations -ContextTypeName
MusicStore.Models.MusicStoreDataContext.
PM> Enable-Migrations -ContextTypeName
MusicStore.Models.MusicStoreDataContext -EnableAutomaticMigrations
```

Solution Explorer:

- Controllers
 - AccountController.cs
 - ArtistsController.cs
 - HomeController.cs
 - ManageController.cs
- fonts
- Models
 - AccountViewModels.cs
 - Artist.cs
 - IdentityModels.cs
 - ManageViewModels.cs
 - MusicStoreDataContext.cs
- obj
- Scripts
- Views
 - Account
 - Artists
 - +Index.cshtml
 - Home
 - Manage
 - Shared
 - +ViewStart.cshtml