



C# Web – MVC

PRO/PRW - C# Web 1

DE HOGESCHOOL MET HET NETWERK

Hogeschool PXL – Elfde-Liniestraat 24 – B-3500 Hasselt
www.pxl.be - www.pxl.be/facebook



Doel

- ASP.Net Core toepassing
 - MVC Web Application
 - Models
 - Views
 - Controllers
 - Entity Framework Core

ASP.Net Core MVC

ENTITY FRAMEWORK

- **MVCFifa**

MVCFifa



Web Application (Model-View-Controller)

A project template for creating an ASP.NET Core application with example ASP.NET Core MVC Views and Controllers. This template can also be used for RESTful HTTP services.

Entity Framework Core – SQL Server – Nuget Package Manager

The image shows the Visual Studio IDE with the **Tools** menu open. The **NuGet Package Manager** option is highlighted, and its sub-menu is visible, showing **Manage NuGet Packages for Solution...** as the selected option.

The **NuGet - Solution** window is open, displaying the **Manage Packages for Solution** interface. The **Browse** tab is active, showing the search results for **entityframeworkcore.SqlServer**. The package **Microsoft.EntityFrameworkCore.SqlServer** is selected, with details including the version **6.0.9** and the **Install** button.

The package details section shows:

- Package name:** Microsoft.EntityFrameworkCore.SqlServer
- Version:** 6.0.9
- Author:** Microsoft
- Description:** Microsoft SQL Server database provider for Entity Framework Core.

The **Installed** tab shows the package is not installed, with the **Install** button highlighted.

- Create Folder Data
 - Add class ApplicationDbContext.cs
 - BaseClass DbContext

```
using Microsoft.EntityFrameworkCore;
```

```
namespace MVCFifa.Data
```

```
{
```

```
    public class ApplicationDbContext : DbContext
```

```
    {
```

```
        public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)  
            : base(options)
```

```
    {
```

```
    }
```

```
}
```

```
}
```

Models folder

- Add Player.cs

```
namespace MVCFifa.Models
{
    public class Player
    {
        public int? PlayerId { get; set; }
        [Required]
        public string? FirstName { get; set; }
        [Required]
        public string? LastName { get; set; }
        public string? ImageLink { get; set; }
    }
}
```

Program.cs

```
using Microsoft.EntityFrameworkCore;
using MVCFifa.Data;
using System.Text;

var builder = WebApplication.CreateBuilder(args);
var sb = new StringBuilder();
sb.Append("Server=(localdb)\\mssqllocaldb;");
sb.Append("Database=fifaDb;");
sb.Append("Trusted_Connection=true;");
sb.Append("MultipleActiveResultSets=true");
var connString = sb.ToString();
builder.Services.AddDbContext<ApplicationDbContext>(
    options => options.UseSqlServer(connString));
```


Controllers folder

- Add PlayerController.cs



Microsoft.EntityFrameworkCore.Design by Microsoft
Shared design-time components for Entity Framework Core tools.

Views folder

- Add folder Player
 - Scaffold Razor View - Index – List Template – Player Model
 - Scaffold Razor View - Create – Create Template – Player Model
 - Geen Data context class selecteren

Add Razor View

View name:

Template:

Model class:

Data context class:

Options:

☐ Create as a partial view

☒ Reference script libraries

☒ Use a layout page:

(Leave empty if it is set in a Razor _viewstart file)

Add

Cancel

Data folder

- Add new Table in database file - ApplicationDbContext

```
public class ApplicationDbContext : DbContext
{
    public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options) : base(options)
    {
    }
    public DbSet<Player>? Players { get; set; }
}
```

Controllers folder

- Add **Player**Controller.cs
 - IActionResult Index()
 - IActionResult Create() (HttpGet)
 - IActionResult Create(Player player) (HttpPost)

Views folder

- Add folder **Player**
 - Razor View – Index.cshtml
 - Razor View – Create.cshtml

Data folder

- DbSet - Table “Players” in database file – ApplicationDbContext

```
public class ApplicationDbContext : DbContext
{
    public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)
        : base(options){}

    public DbSet<Player>? Players { get; set; }
}
```

Controllers - PlayerController

```
public class PlayerController : Controller
{
    public IActionResult Index()
    {
        return View();
    }
    [HttpGet]
    public IActionResult Create()
    {
        return View();
    }
    [HttpPost]
    public IActionResult Create(Player player)
    {
        return View();
    }
}
```

Dependency Injection

- PlayerController
 - Add Constructor
 - PlayerController(DbContext Service)

```
public class PlayerController : Controller
{
    ApplicationDbContext _context;
    public PlayerController(ApplicationDbContext context)
    {
        _context = context;
        _context.Database.EnsureCreated();
    }
    ...
}
```

- Styling the content
 - Views/Shared
 - _Layout.cshtml

- wwwroot folder
 - Add folder images
 - Add image fifa...

```
<header>
  <div class="text-center">
    
  </div>
</header>
<div class="row">
  <div class="col-1"></div>
  <div class="col-3">
    <br />
    <a asp-controller="Player" asp-action="Index">Players</a>
  </div>
  <div class="col-8"> @RenderBody() </div>
</div>
<footer class="border-top footer text-muted">
  <div class="container">
    &copy; PXL- MVCFifa
  </div>
</footer>
```

- PlayerController
 - Read data
 - Index

```
public IActionResult Index()  
{  
    var players = _context.Players;  
    return View(players);  
}
```

- PlayerController
 - Insert data
 - Create

```
[HttpGet]
public IActionResult Create()
{
    var player = new Player();
    return View(player);
}
[HttpPost]
public IActionResult Create(Player player)
{
    if (ModelState.IsValid)
    {
        AddPlayer(player);
        return RedirectToAction("Index");
    }
    return View(player);
}
private void AddPlayer(Player player)
{
    _context.Players.Add(player);
    _context.SaveChanges();
}
```

Samenvatting Entity Framework

- Install Nuget package
- Update Startup class Program.cs
 - Service DbContext
- Create DbContext class file in Data folder
- Create Model -> Table -> DbSet
- Dependency Injection
 - Context class -> Controller
 - Context.SaveChanges()

Oefening

- Edit/Details/Delete

Add Razor View

View name:

Template:

Model class:

Data context class:

Options:

☐ Create as a partial view

☒ Reference script libraries

☒ Use a layout page:

...

(Leave empty if it is set in a Razor _viewstart file)

Add Cancel

Views folder

- Scaffold razor views in Player folder
 - Scaffold Razor View - Details – Details Template – Player Model
 - Scaffold Razor View - Edit – Edit Template – Player Model
 - Scaffold Razor View - Delete – Delete Template – Player Model

- PlayerController
 - Details

```
public IActionResult Details(int id)
{
    var player = _context.Players.Where(x=>x.PlayerId == id).FirstOrDefault();
    return View(player);
}
```

Views/Player/Index.cshtml

```
...
@Html.ActionLink("Edit", "Edit", new { id = item.PlayerId }) |
@Html.ActionLink("Details", "Details", new { id=item.PlayerId}) |
@Html.ActionLink("Delete", "Delete", new { id = item.PlayerId })
<br>
<a asp-action="Details" asp-route-id="item.PlayerID">Details</a>
```

- PlayerController
 - Delete

```
[HttpGet]
public IActionResult Delete(int id)
{
    var player = _context.Players.Where(x => x.PlayerId == id).FirstOrDefault();
    return View(player);
}
[HttpPost]
public IActionResult DeletePost(int id)
{
    var player = _context.Players.Where(x => x.PlayerId == id).FirstOrDefault();
    _context.Players.Remove(player);
    _context.SaveChanges();
    return RedirectToAction("Index");
}
```

Views/Player/Delete.cshtml

```
<form asp-action="DeletePost" asp-route-id="@Model.PlayerId">
    <input type="submit" value="Delete" class="btn btn-danger" /> |
    <a asp-action="Index">Back to List</a>
</form>
```

- PlayerController
 - Update data
 - Edit

```
[HttpGet]
public IActionResult Edit(int id)
{
    var player = _context.Players
        .Where(x => x.PlayerId == id)
        .FirstOrDefault();
    return View(player);
}

[HttpPost]
public IActionResult Edit(Player player)
{
    if (ModelState.IsValid)
    {
        UpdatePlayer(player);
        return RedirectToAction("Index");
    }
    return View(player);
}

private void UpdatePlayer(Player player)
{
    _context.Players.Update(player);
    _context.SaveChanges();
}
```