C# Web – MVC

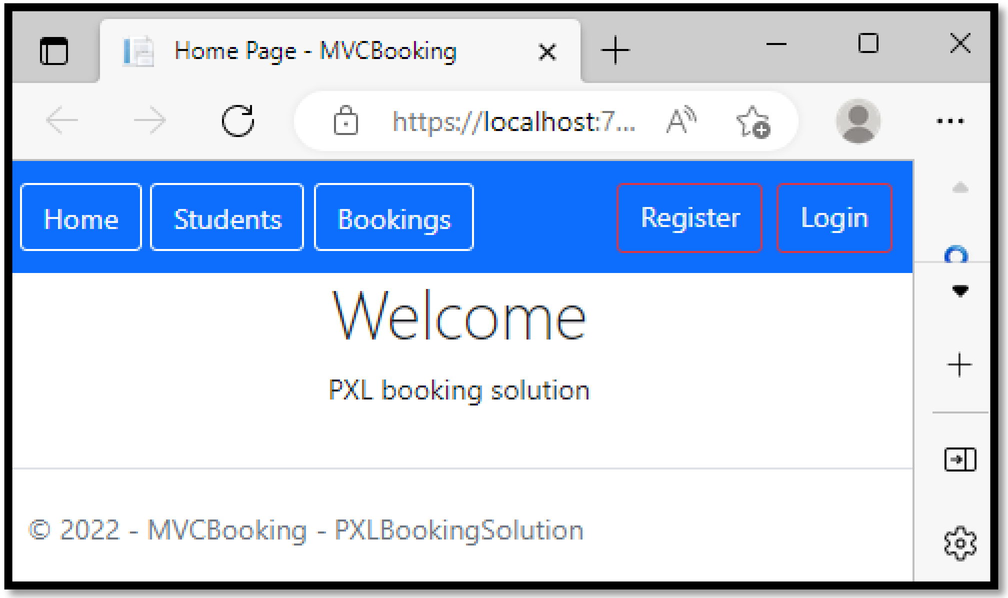
PRO/PRW - C# Web 1

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
  - Identity Framework
    - Custom Identity User class (Student)
    - Identity Role
    - Identity Options



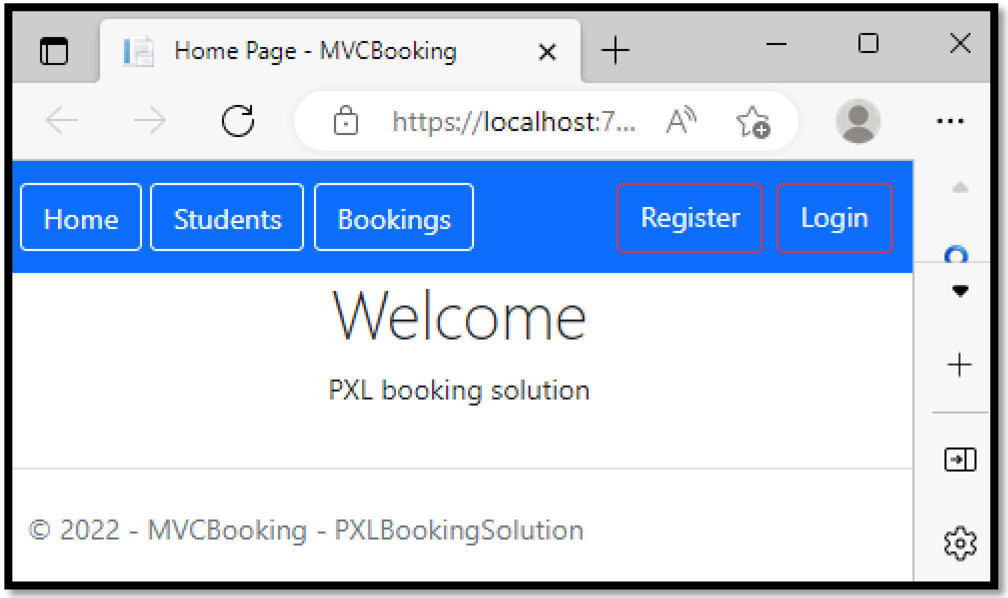
Welcome

PXL booking solution

© 2022 - MVCBooking - PXLBookingSolution

Startbestand

- Blackboard
  - – Identity Framework
    - BookingSolution.zip



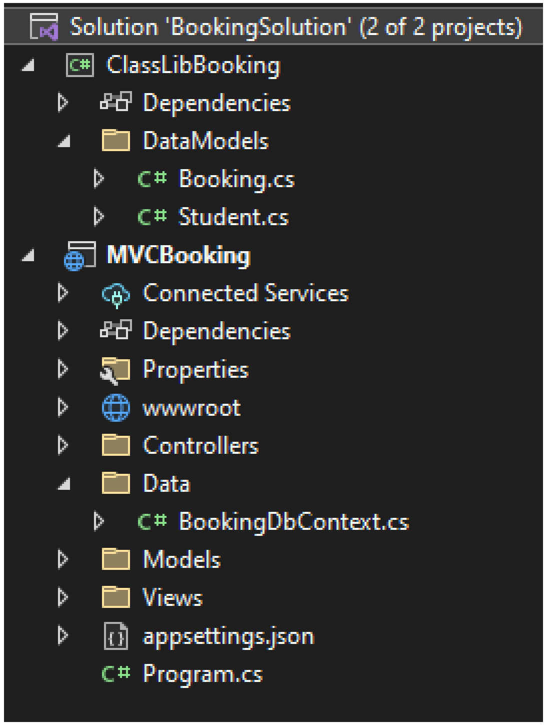
Welcome

PXL booking solution

© 2022 - MVCBooking - PXLBookingSolution

Architectuur

- Architectuur
  - Class Library
    - Data models
      - Booking
      - Student
  - Model View Controller Web App
    - Data
      - BookingDbContext



```
Solution 'BookingSolution' (2 of 2 projects)
  ClassLibBooking
    Dependencies
    DataModels
      Booking.cs
      Student.cs
  MVCBooking
    Connected Services
    Dependencies
    Properties
    wwwroot
    Controllers
    Data
      BookingDbContext.cs
    Models
    Views
    appsettings.json
    Program.cs
```

Class Library

- MVCBooking
  - Add reference -> ClassLibBooking
- Nuget Package Manager
  - Class Library
    - Microsoft.AspNetCore.Identity.EFCore

Class Library

DataModels/Student.cs

```
namespace ClassLibBooking.DataModels
{
    public class Student : IdentityUser
    {
        public int StudentId { get; set; }
        [Required]
        public string FirstName { get; set; }
        [Required]
        public string LastName { get; set; }
        public ICollection<Booking>? Bookings { get; set; }
    }
}
```

Data - BookingDbContext

```
namespace MVCBooking.Data
{
    public class BookingDbContext : IdentityDbContext<Student>
    {
        public BookingDbContext(DbContextOptions<BookingDbContext> options)
                : base(options)
        { }
    }
}
```

Program.cs

```
Replace IdentityUser -> Student (base class IdentityUser)
…
builder.Services.AddIdentity<Student, IdentityRole>()
    .AddEntityFrameworkStores<BookingDbContext>();
…
app.UseRouting();

app.UseAuthentication();
app.UseAuthorization();

app.MapControllerRoute( …
```

Views/Shared/_NavbarPartial

```
@using ClassLibBooking.DataModels;
@using Microsoft.AspNetCore.Identity
@inject SignInManager<Student> SignInManager
```

AccountController - Register

```
#region register
[HttpGet]
public IActionResult Register()
{
    return View();
}

[HttpPost]
public IActionResult Register(RegisterViewModel registerViewModel)
{
    if (ModelState.IsValid)
       return View("Login");
    return View();
}
#endregion
```

AccountController - Register

```
[HttpPost]
public IActionResult Register(RegisterViewModel registerViewModel)
{
    if (ModelState.IsValid)
        if (CreateUser(registerViewModel) == IdentityResult.Success)
            return View("Login");
    return View();
}
private IdentityResult CreateUser(RegisterViewModel registerViewModel)
{
    return IdentityResult.Success;
}
private void Errors(IdentityResult identityResult)
{
    foreach (var error in identityResult.Errors)
    {
        ModelState.AddModelError("", error.Description);
    }
}
```

AccountController
Dependency Injection
UserManager<Student>

```
public class AccountController : Controller
{
    UserManager<Student> _userManager;
    public AccountController(UserManager<Student> userManager)
    {
        _userManager = userManager;
    }
    …
```

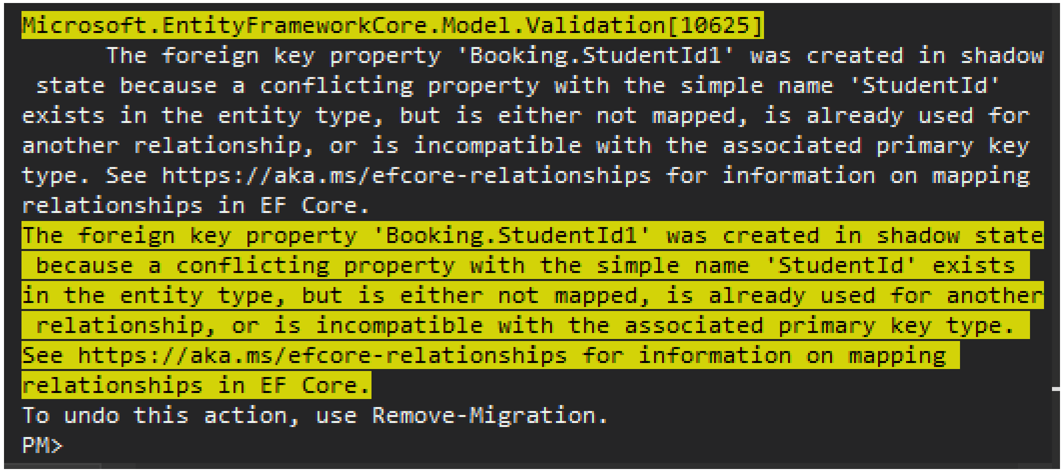
AccountController - Register

```
private async Task<IdentityResult> CreateUserAsync(
                                            RegisterViewModel registerViewModel)
{
    var student = new Student();
    student.FirstName = registerViewModel.FirstName;
    student.LastName = registerViewModel.LastName;
    student.Email = registerViewModel.Email;
    student.UserName = registerViewModel.UserName;
    return await _userManager.CreateAsync(student, registerViewModel.Password);
}
```

```
public async Task<IActionResult> RegisterAsync(RegisterViewModel registerViewModel)
{
    if (ModelState.IsValid)
    {
        var result = await CreateUserAsync(registerViewModel);
        if (result == IdentityResult.Success)
            return View("Login");
        else
            Errors(result);
    }
    return View();
}
```

Package Manager Console

- Add-migration initial



```
Microsoft.EntityFrameworkCore.Model.Validation[10625]
      The foreign key property 'Booking.StudentId1' was created in shadow
 state because a conflicting property with the simple name 'StudentId'
exists in the entity type, but is either not mapped, is already used for
another relationship, or is incompatible with the associated primary key
type. See https://aka.ms/efcore-relationships for information on mapping
relationships in EF Core.
The foreign key property 'Booking.StudentId1' was created in shadow state
 because a conflicting property with the simple name 'StudentId' exists
in the entity type, but is either not mapped, is already used for another
 relationship, or is incompatible with the associated primary key type.
See https://aka.ms/efcore-relationships for information on mapping
relationships in EF Core.
To undo this action, use Remove-Migration.
PM>
```

Refactor - Student

ClassLibBooking – DataModels/Student.cs

- Remove StudentId (IdentityUser.Id) -> (AspNetUsers.Id)

```
namespace ClassLibBooking.DataModels
{
    public class Student : IdentityUser
    {
        [Required]
        public string FirstName { get; set; }
        [Required]
        public string LastName { get; set; }
        public ICollection<Booking>? Bookings { get; set; }
    }
}
```

Refactor - Booking

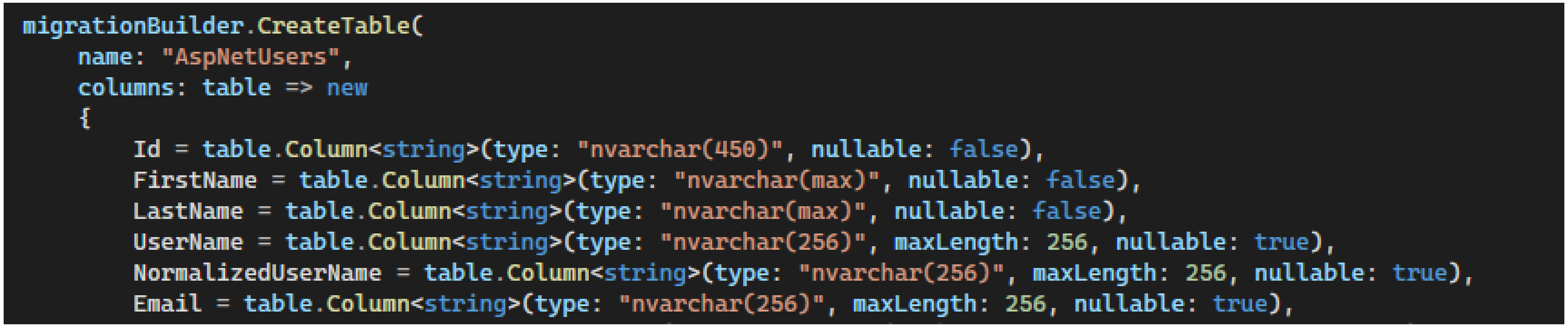
ClassLibBooking – DataModels/Booking.cs

- Change StudentId datatype (-> AspNetUsers.Id)

```
namespace ClassLibBooking.DataModels
{
    public class Booking
    {
        public int BookingId { get; set; }
        [Required]
        public string StudentId { get; set; }
        [Required]
        public DateTime BookingDate { get; set; }
        public string? Remarks { get; set; }
        public Student? Student { get; set; }
    }
}
```

Package Manager Console

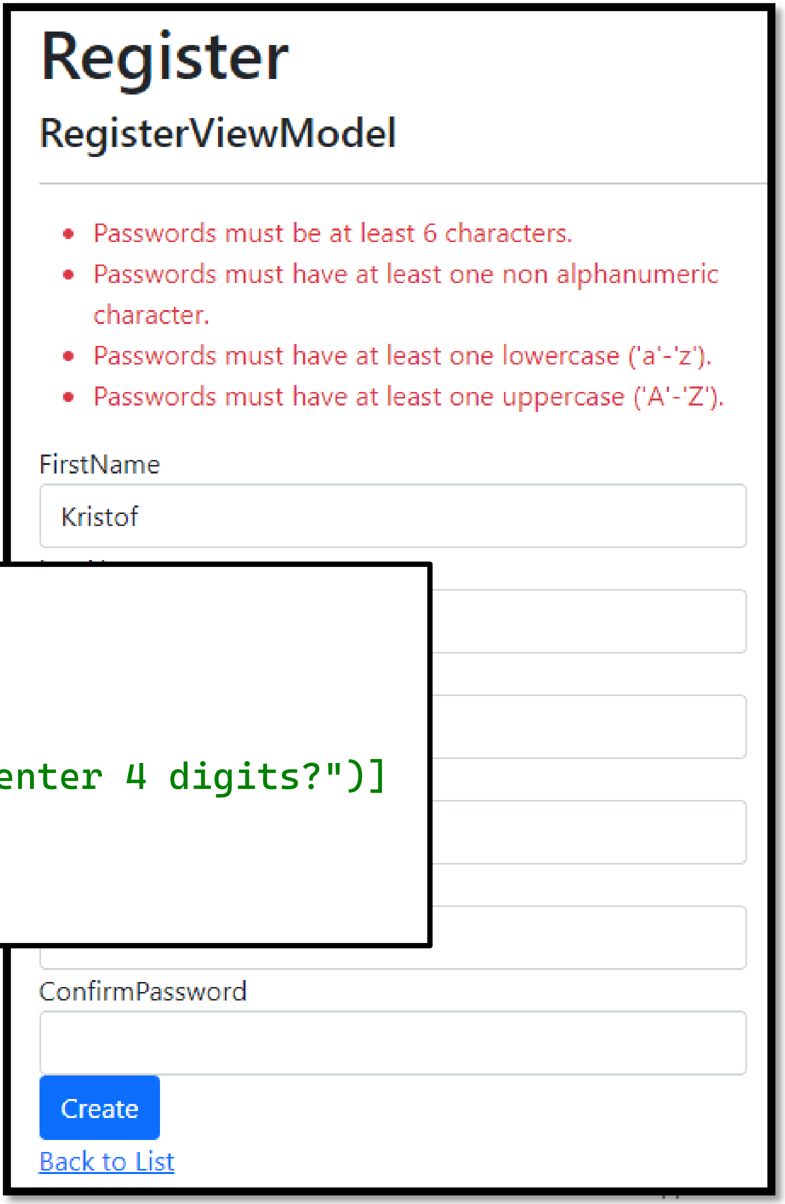
- Remove-migration
- Add-migration initial



```
migrationBuilder.CreateTable(
    name: "AspNetUsers",
    columns: table => new
    {
        Id = table.Column<string>(type: "nvarchar(450)", nullable: false),
        FirstName = table.Column<string>(type: "nvarchar(max)", nullable: false),
        LastName = table.Column<string>(type: "nvarchar(max)", nullable: false),
        UserName = table.Column<string>(type: "nvarchar(256)", maxLength: 256, nullable: true),
        NormalizedUserName = table.Column<string>(type: "nvarchar(256)", maxLength: 256, nullable: true),
        Email = table.Column<string>(type: "nvarchar(256)", maxLength: 256, nullable: true),
```

- Update-database

AccountController - Register



Register

RegisterViewModel

- Passwords must be at least 6 characters.
- Passwords must have at least one non alphanumeric character.
- Passwords must have at least one lowercase ('a'-'z').
- Passwords must have at least one uppercase ('A'-'Z').

FirstName

Kristof

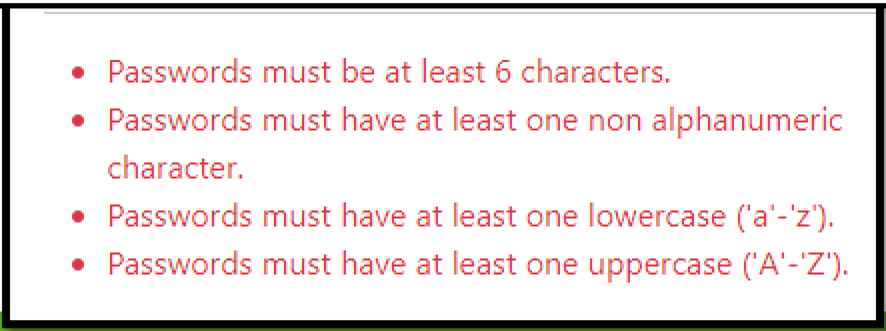
ConfirmPassword

```
public class RegisterViewModel
{
    ...
    [Required]
    // [Display(Description = "Password: please enter 4 digits?")]
    // [Range(0, 9999)]
    [DataType(DataType.Password)]
    public string Password { get; set; }
```

Identity Options

```
builder.Services.AddIdentity<Student, IdentityRole>(passwordOptions=>
    {
        passwordOptions.Password.RequireNonAlphanumeric = false;
        passwordOptions.Password.RequireUppercase = false;
        passwordOptions.Password.RequireLowercase = false;
        passwordOptions.Password.RequiredLength = 4;
    }).AddEntityFrameworkStores<BookingDbContext>();
```

```
[Required]
[EmailAddress]
public string Email { get; set; }
[Required]
[Display(Description = "Password: please enter 4 digits?")]
[Range(0, 9999)]
[DataType(DataType.Password)]
public string Password { get; set; }
[DataType(DataType.Password)]
[Compare("Password", ErrorMessage = "Password doesn't match!")]
public string ConfirmPassword { get; set; }
```



- Passwords must be at least 6 characters.
- Passwords must have at least one non alphanumeric character.
- Passwords must have at least one lowercase ('a'-'z').
- Passwords must have at least one uppercase ('A'-'Z').

Identity Options

```
builder.Services.AddIdentity<Student, IdentityRole>
 (passwordOptions=>
   {
    passwordOptions.Password.RequireNonAlphanumeric = false;
    passwordOptions.Password.RequireUppercase = false;
    passwordOptions.Password.RequireLowercase = false;
    passwordOptions.Password.RequiredLength = 4;
   }
  )
    .AddEntityFrameworkStores<BookingDbContext>();
```

```
builder.Services.Configure<IdentityOptions>
(options =>
  {
      // Password settings.
      options.Password.RequireDigit = true;
      …
  }
 );
```

Oefening

- StudentController
  - Enkel toegankelijk als men is ingelogd
  - Index – List enige action

- BookingController
  - Enkel toegankelijk als men is ingelogd
  - Create action – de user die is ingelogd staat hardcoded in input form field van de identity user

- AccountController
  - Login actie
  - Logout actie

- Navbar
  - Valideer hyperlinks

- Wijzig Identity Options
  - Verplicht 5 digits