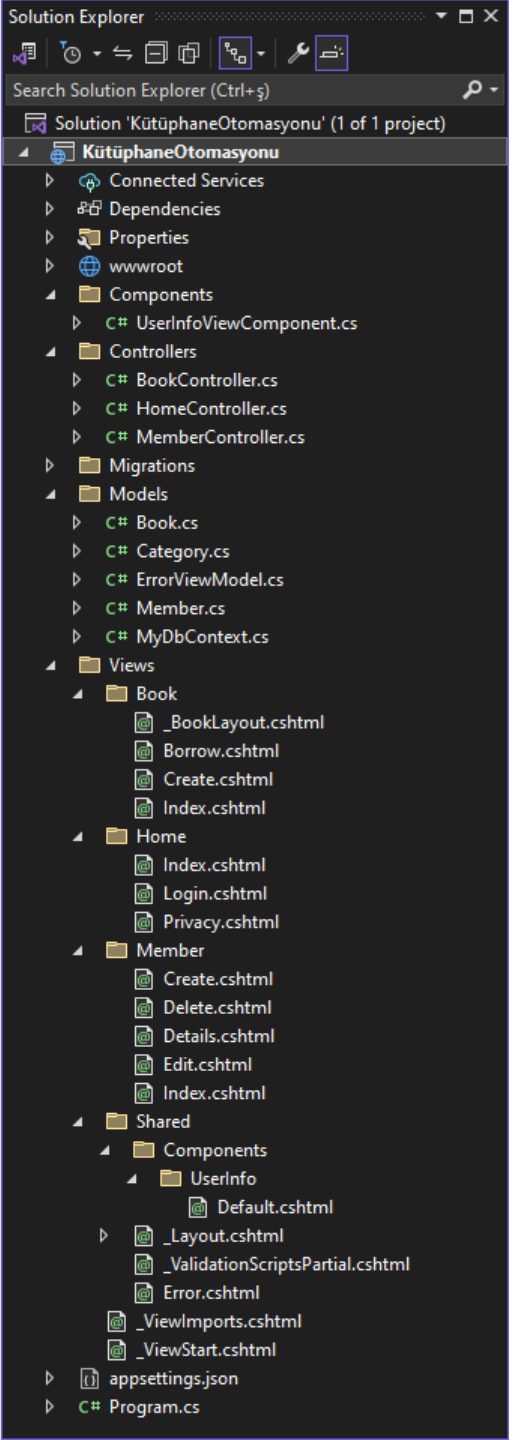
KÜTÜPHANE PROJESİ

Orhan SARIÇİÇEK

Muhammed Bedir KÜÇÜKHAZAR

Hakan KARATEKE



Kütüphane Otomasyonu

Models

- Book.cs
- Member.cs
- Category.cs
- MyDbContext.cs
- ErrorViewModel.cs

Controllers

- HomeController.cs
- BookController.cs
- MemberController.cs

Views

- Home
  - Index.cshtml
  - Login.cshtml
  - Privacy.cshtml
- Book
  - _BookLayout.cshtm
  - Borrow.cshtml
  - Create.cshtml
  - Index.cshtml
- Member
  - Create.cshtml
  - Details.cshtml
  - Delete.cshtml
  - Edit.cshtml
  - Index.cshtml

```
// Add services to the container.
builder.Services.AddControllersWithViews();
builder.Services.AddDbContext<MyDbContext>(options =>
    options.UseSqlServer(builder.Configuration.GetConnectionString("ConStr")));
builder.Services.AddScoped<MyDbContext>();
```

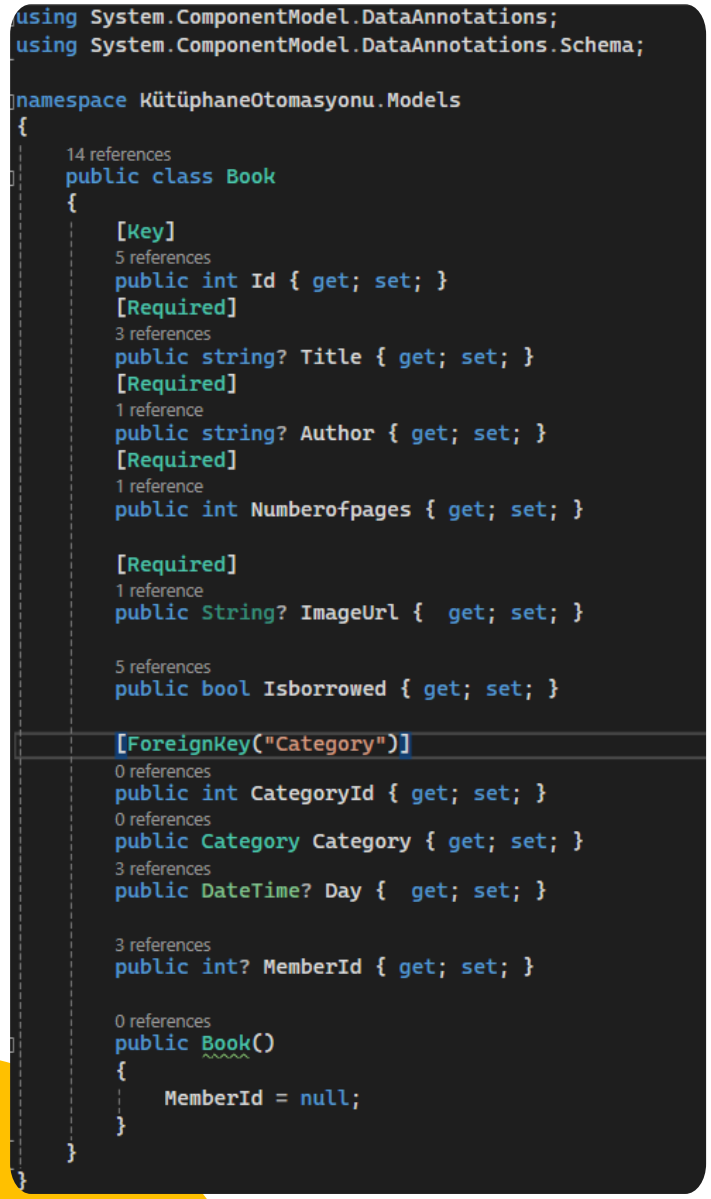
```
{
"Logging": {
  "LogLevel": {
    "Default": "Information",
    "Microsoft.AspNetCore": "Warning"
  }
},
"AllowedHosts": "*",
"ConnectionStrings": {
  "ConStr": "Server=.;Database=libraryDB;Trusted_Connection=SSPI;TrustServerCertificate=true;"
}
```

SQL Bağlantı işlemleri

Burada gösterilen kodlar ile sql bağlantısı sağlanmıştır

Modeller

Projede kullanılan sınıflar bu bölümde anlatılmıştır



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

namespace KütüphaneOtomasyonu.Models
{
    14 references
    public class Book
    {
        [Key]
        5 references
        public int Id { get; set; }
        [Required]
        3 references
        public string? Title { get; set; }
        [Required]
        1 reference
        public string? Author { get; set; }
        [Required]
        1 reference
        public int Numberofpages { get; set; }

        [Required]
        1 reference
        public String? ImageUrl {  get; set; }

        5 references
        public bool Isborrowed { get; set; }

        [ForeignKey("Category")]
        0 references
        public int CategoryId { get; set; }
        0 references
        public Category Category { get; set; }
        3 references
        public DateTime? Day {  get; set; }

        3 references
        public int? MemberId { get; set; }

        0 references
        public Book()
        {
            MemberId = null;
        }
    }
}
```

Book.cs

- Değişkenler:
- **Id: (int, [Key])** Kitabın benzersiz kimlik numarasını temsil eder.
- **Title: (string?, [Required])** Kitabın başlığını temsil eder.
- **Author: (string?, [Required])** Kitabın yazarını temsil eder.
- **Numberofpages: (int, [Required])** Kitabın sayfa sayısını temsil eder.
- **ImageUrl: (string?, [Required])** Kitabın kapak resminin URL'sini temsil eder.
- **Isborrowed: (bool)** Kitabın ödünç alınıp alınmadığını gösterir.
- **CategoryId: (int, [ForeignKey("Category")])** Kitabın kategorisinin kimlik numarasını temsil eder.
- **Category: (Category)** Kitabın kategorisini temsil eden nesne.
- **Day: (DateTime?)** Kitabın ödünç alındığı tarihi temsil eder.
- **MemberId: (int?)** Kitabı ödünç alan üyenin kimlik numarasını temsil eder.
- Metotlar:
- **Constructor: (public Book())** Üye kimliği alanını ilk değer olarak null olarak ayarlayan yapıcıdır.

Category.cs

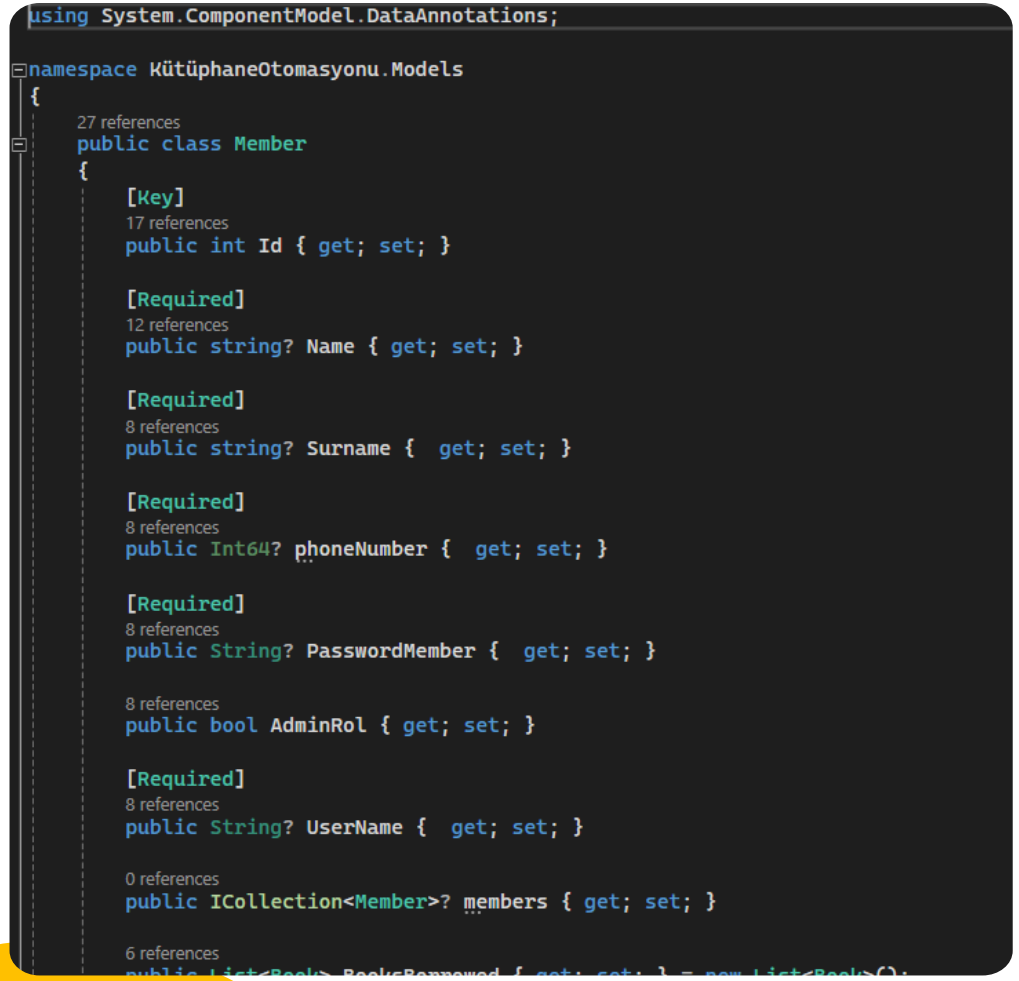
```
using System.ComponentModel.DataAnnotations;

namespace KütüphaneOtomasyonu.Models
{
    3 references
    public class Category
    {

        0 references
        public int Id { get; set; }
        0 references
        public string? Name { get; set; }

        0 references
        public ICollection<Category> Categories { get; set; }
    }
}
```

Değişkenler:

- **Id: (int)** Kategorinin benzersiz kimlik numarasını temsil eder.
- **Name: (string?)** Kategorinin adını temsil eder.
- **Categories: (ICollection<Category>)** Alt kategorileri temsil eden bir koleksiyondur. Bu, kategorilerin hiyerarşik bir yapıya sahip olabileceğini gösterir.
- Metotlar:
- **Get ve Set Metotları:** Otomatik olarak oluşturulan, değişkenlerin değerlerini okumak ve yazmak için kullanılan metotlardır.

DataAnnotations: Doğrulama ve veri modelleme özellikleri sağlamak için kullanılan bir namespace'dir.

- **Not: Bu model, kategorilerin hiyerarşik bir yapıya sahip olabileceğini gösterir. Örneğin, "Bilim Kurgu" kategorisinin "Uzay Operası" ve "Siberpunk" gibi alt kategorileri olabilir.**



```
using System.ComponentModel.DataAnnotations;

namespace KütüphaneOtomasyonu.Models
{
    27 references
    public class Member
    {
        [Key]
        17 references
        public int Id { get; set; }

        [Required]
        12 references
        public string? Name { get; set; }

        [Required]
        8 references
        public string? Surname {  get; set; }

        [Required]
        8 references
        public Int64? phoneNumber {  get; set; }

        [Required]
        8 references
        public String? PasswordMember {  get; set; }

        8 references
        public bool AdminRol { get; set; }

        [Required]
        8 references
        public String? UserName {  get; set; }

        0 references
        public ICollection<Member>? members { get; set; }

        6 references
```

Member.cs

Değişkenler:

- **Id: (int, [Key])** Üyenin benzersiz kimlik numarasını temsil eder.
- **Name: (string?, [Required])** Üyenin adını temsil eder.
- **Surname: (string?, [Required])** Üyenin soyadını temsil eder.
- **phoneNumber: (Int64?, [Required])** Üyenin telefon numarasını temsil eder.
- **PasswordMember: (string?, [Required])** Üyenin şifresini temsil eder.
- **AdminRol: (bool)** Üyenin yönetici rolüne sahip olup olmadığını gösterir.
- **UserName: (string?, [Required])** Üyenin kullanıcı adını temsil eder.
- **members: (ICollection<Member>?)** Üyenin alt üyelerini temsil eden bir koleksiyondur. Bu, üyelerin hiyerarşik bir yapıya sahip olabileceğini gösterir.
- **BooksBorrowed: (List<Book>)** Üyenin ödünç aldığı kitapları temsil eden bir listedir.

Metotlar:

- **Get ve Set Metotları**: Otomatik olarak oluşturulan, değişkenlerin değerlerini okumak ve yazmak için kullanılan metotlardır.

MyDbContext.cs

```
using Microsoft.EntityFrameworkCore;

namespace KütüphaneOtomasyonu.Models
{
    28 references
    public class MyDbContext  : DbContext
    {
        0 references
        public MyDbContext(DbContextOptions<MyDbContext> o) ...
        1 reference
        public Member CurrentUser { get; set; }
        5 references
        public DbSet<Book> Books { get; set; }
        10 references
        public DbSet<Member> Members { get; set; }
        0 references
        public DbSet<Category> Categories { get; set; }


    }
}
```

Değişkenler:

- **CurrentUser: (Member)** Oturum açmış olan üyeyi temsil eden nesne.
- **Books: (DbSet<Book>)** Kitapların koleksiyonunu temsil eder.
- **Members: (DbSet<Member>)** Üyelerin koleksiyonunu temsil eder.
- **Categories: (DbSet<Category>)** Kategorilerin koleksiyonunu temsil eder.
- **Metotlar**:
- **Yapıcı: (public MyDbContext(DbContextOptions<MyDbContext> o) : base(o))** Veritabanı bağlantı seçeneklerini kullanarak veritabanına bağlantı kurar.

DbContext: Entity Framework Core'da veritabanı işlemlerini yönetmek için kullanılan temel sınıftır.

- ***DbSet: Veritabanında belirli bir modelin koleksiyonunu temsil eden bir sınıftır.***
- ***CurrentUser: Bu değişken, veritabanına yapılan sorgularda ve işlemlerde geçerli kullanıcının bilgilerinin kullanılmasını sağlar.***
- ***Bu DbContext sınıfı, projedeki modeller ile veritabanı arasındaki köprü rolünü üstlenir.***

Controller Yapıları

ASP.NET Core MVC'de web isteklerini işleyen ve yanıtlar oluşturan sınıfların temel sınıfıdır.
Projede kullanılan Controller yapıları bu bölümde anlatılmıştır

```
0 references
public class HomeController : Controller
{
    private readonly ILogger<HomeController> _logger;
    private readonly MyDbContext _context;

    public static Member Memberdef=new Member();

    0 references
    public HomeController(ILogger<HomeController> logger, MyDbContext context)...

    0 references
    public IActionResult Index()...

    0 references
    public IActionResult Privacy()...

    0 references
    public IActionResult Login()...

    [HttpPost]
    0 references
    public IActionResult Login(string username, string password)
    {
        // Kullanıcı giriş bilgilerini kontrol et (örneğin, veritabanında kontrol edebilirsiniz)

        var members = _context.Members.ToList();
        var books = _context.Books.ToList();

        Member user = null;
        foreach (var u in members)...
        if (user != null)
        {
            // Kullanıcı girişi başarılı ise CurrentUser'ı ayarla
            _context.CurrentUser = user;

            HttpContext.Items["CurrentUser"] = user;
            // Kullanıcı girişi başarılı ise BookController'ın Index eylemine yönlendir
            if (user.AdminRol)
            {
                return RedirectToAction("Index", "Member");
            }

            return RedirectToAction("Index", "Book");
        }

        // Giriş başarısız ise aynı sayfaya geri dön
        return View();
    }
```

HomeController.cs

Metotlar:

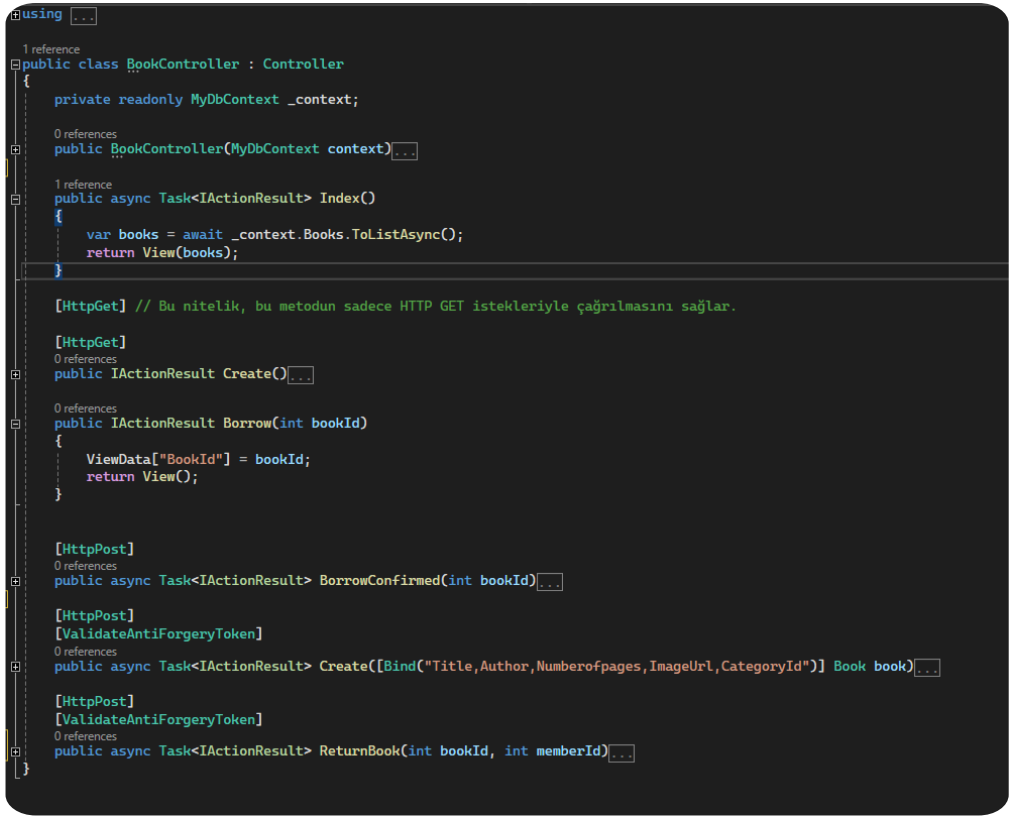
- **Yapıcı: (public HomeController(ILogger<HomeController> logger, MyDbContext context))** Günlükleyici ve veritabanı bağlamını başlatır.
- **Index:** Ana sayfa görünümünü döndürür.
- **Privacy:** Gizlilik politikası görünümünü döndürür.
- **Login:** Giriş sayfası görünümünü döndürür.
- **Login (HttpPost):** Kullanıcı giriş bilgilerini kontrol eder ve başarılı olursa kullanıcıyı uygun sayfaya yönlendirir:
    - *Veritabanından üyeleri ve kitapları sorgular.*
    - *Kullanıcı adını ve şifreyi veritabanındaki bilgilerle karşılaştırır.*
    - *Giriş başarılı olursa, kullanıcının ödünç aldığı kitapları BooksBorrowed özelliğine ekler.*
    - *CurrentUser değişkenini ayarlar ve kullanıcıyı uygun sayfaya yönlendirir.*

İlave Açıklamalar:

- **IActionResult: Bir eylemin döndürebileceği çeşitli sonuç türlerini temsil eden bir arabirimdir.**
- **HttpPost: Bir form gönderimini işleyen eylemleri belirtmek için kullanılan bir özniteliktir.**
- **ResponseCache: Bir eylemin yanıtının önbelleğe alınmasını denetlemek için kullanılan bir özniteliktir.**
- **ErrorViewModel: Hata bilgilerini içeren bir görünüm modelini temsil eder.**

Bu HomeController sınıfı, uygulamanın giriş ve ana sayfa işlevselliğini yönetir.

BookController.cs



```
using ...

1 reference
public class BookController : Controller
{
    private readonly MyDbContext _context;

    0 references
    public BookController(MyDbContext context)...

    1 reference
    public async Task<IActionResult> Index()
    {
        var books = await _context.Books.ToListAsync();
        return View(books);
    }

    [HttpGet] // Bu nitelik, bu metodun sadece HTTP GET istekleriyle çağrılmasını sağlar.

    [HttpGet]
    0 references
    public IActionResult Create()...

    0 references
    public IActionResult Borrow(int bookId)
    {
        ViewData["BookId"] = bookId;
        return View();
    }


    [HttpPost]
    0 references
    public async Task<IActionResult> BorrowConfirmed(int bookId)...

    [HttpPost]
    [ValidateAntiForgeryToken]
    0 references
    public async Task<IActionResult> Create([Bind("Title,Author,Numberofpages,ImageUrl,CategoryId")] Book book)...

    [HttpPost]
    [ValidateAntiForgeryToken]
    0 references
    public async Task<IActionResult> ReturnBook(int bookId, int memberId)...
}
```

- Yapıcı: (public BookController(MyDbContext context)) Veritabanı bağlamını başlatır.
- Index: Tüm kitapların listesini döndürür.
- Create: Yeni bir kitap eklemek için bir form görünümünü döndürür.
- Borrow: Bir kitabı ödünç almak için bir onay sayfası görünümünü döndürür.

BookController.cs

BorrowConfirmed (HttpPost): Kitabı ödünç alma işlemini gerçekleştirir:

- Kitabı veritabanından bulur.
- Kitap ödünç alınmışsa hata mesajı gösterir.
- Kitap ödünç alınmamışsa, kitap bilgilerini günceller (Isborrowed, MemberId, Day) ve veritabanını kaydeder.
- Üyenin detay sayfasına yönlendirir.

Create (HttpPost): Yeni bir kitap oluşturma işlemini gerçekleştirir:

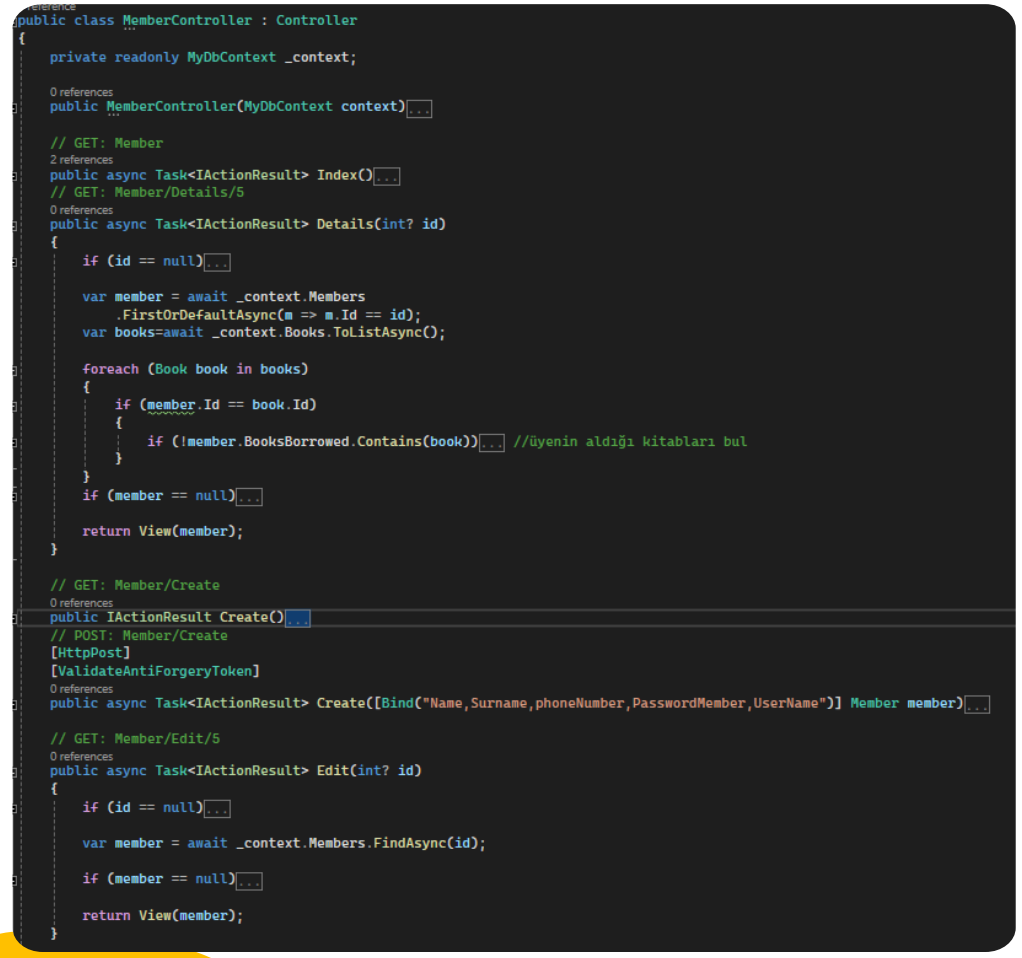
- Formdan gelen kitap bilgilerini alır.
- Varsayılan değerleri atar (Isborrowed = false).
- Kitabı veritabanına ekler ve veritabanını kaydeder.
- Kitap listesine yönlendirir.

ReturnBook (HttpPost): Bir kitabı iade etme işlemini gerçekleştirir:

- Kitap ve üye bilgilerini veritabanından bulur.
- Kitap veya üye bulunamazsa hata mesajı gösterir.
- Kitap bilgilerini günceller (Isborrowed, MemberId, Day) ve veritabanını kaydeder.
- Üyenin detay sayfasına yönlendirir.

```
    if (book == null)...

    if (book.Isborrowed)...
    else
    {
        // Kitabı ödünç al
        book.Isborrowed = true;
        book.MemberId = HomeController.Memberdef.Id;
        book.Day = DateTime.Now;

        _context.Update(book);
        await _context.SaveChangesAsync();
    }

    // Member controller'ındaki Details view sayfasına yönlendir
    return RedirectToAction("Details", "Member", HomeController.Memberdef);
}

[HttpPost]
[ValidateAntiForgeryToken]
0 references
public async Task<IActionResult> Create([Bind("Title,Author,Numberofpages,ImageUrl,CategoryId")] Book book)
{
    // Default değerler atanabilir
    book.Isborrowed = false; // Varsayılan olarak kitap ödünç alınmamış olarak ayarlanır.

        _context.Add(book);
        await _context.SaveChangesAsync();
        return RedirectToAction(nameof(Index));
}

[HttpPost]
[ValidateAntiForgeryToken]
0 references
public async Task<IActionResult> ReturnBook(int bookId, int memberId)
{
    // Kitap ve üye var mı diye kontrol et
    var book = await _context.Books.FindAsync(bookId);
    var member = await _context.Members.FindAsync(memberId);

    if (book == null || member == null)
    {
        return NotFound();
    }

    // Kitabı iade et
    book.Isborrowed = false;
    book.MemberId = null;
    book.Day = null;

    _context.Update(book);
    await _context.SaveChangesAsync();

    // Üye detay sayfasına yönlendir
    return RedirectToAction("Details", "Member", new { id = memberId });
}
```

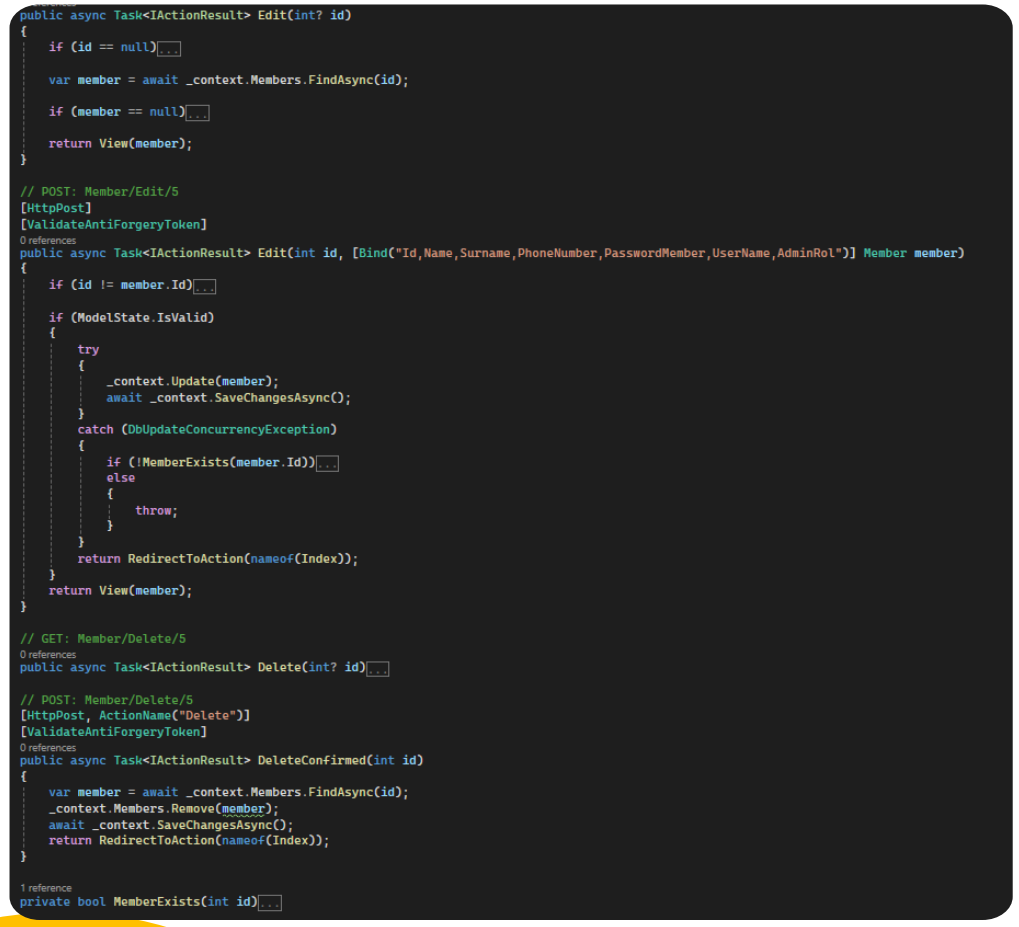
MemberController.cs



```
public class MemberController : Controller
{
    private readonly MyDbContext _context;

    0 references
    public MemberController(MyDbContext context)...

    // GET: Member
    2 references
    public async Task<IActionResult> Index()...
    // GET: Member/Details/5
    0 references
    public async Task<IActionResult> Details(int? id)
    {
        if (id == null)...

        var member = await _context.Members
            .FirstOrDefaultAsync(m => m.Id == id);
        var books=await _context.Books.ToListAsync();

        foreach (Book book in books)
        {
            if (member.Id == book.Id)
            {
                if (!member.BooksBorrowed.Contains(book))... //üyenin aldığı kitabları bul
            }
        }
        if (member == null)...

        return View(member);
    }

    // GET: Member/Create
    0 references
    public IActionResult Create()...
    // POST: Member/Create
    [HttpPost]
    [ValidateAntiForgeryToken]
    0 references
    public async Task<IActionResult> Create([Bind("Name,Surname,phoneNumber,PasswordMember,UserName")] Member member)...

    // GET: Member/Edit/5
    0 references
    public async Task<IActionResult> Edit(int? id)
    {
        if (id == null)...

        var member = await _context.Members.FindAsync(id);

        if (member == null)...

        return View(member);
    }
```

1. **Index():**
    1. Veritabanında bulunan tüm üye listesini getirir.
    2. Listeyi görüntülemek için bir görünüm oluşturur.
2. **Details(int? id):**
    1. Üyenin ID'sine göre belirli bir üyenin ayrıntılarını getirir.
    2. Üye tarafından ödünç alınan kitapların bir listesini getirir.
    3. Üyenin ayrıntılarını ve ödünç alınan kitaplarını görüntülemek için bir görünüm oluşturur.
3. Create():
    1. Yeni bir üye oluşturmak için bir görünüm oluşturur.
4. **Create([Bind("Name,Surname,phoneNumber,PasswordMember,UserName")] Member member):**
    1. Yeni bir üye oluşturma isteğini işler.
    2. Üye verilerini doğrular.
    3. Üyeyi veritabanına ekler.
    4. Giriş sayfasına yönlendirir.

MemberController.cs



```
public async Task<IActionResult> Edit(int? id)
{
    if (id == null)...

    var member = await _context.Members.FindAsync(id);

    if (member == null)...

    return View(member);
}

// POST: Member/Edit/5
[HttpPost]
[ValidateAntiForgeryToken]
0 references
public async Task<IActionResult> Edit(int id, [Bind("Id,Name,Surname,PhoneNumber,PasswordMember,UserName,AdminRol")] Member member)
{
    if (id != member.Id)...

    if (ModelState.IsValid)
    {
        try
        {
            _context.Update(member);
            await _context.SaveChangesAsync();
        }
        catch (DbUpdateConcurrencyException)
        {
            if (!MemberExists(member.Id))...
            else
            {
                throw;
            }
        }
        return RedirectToAction(nameof(Index));
    }
    return View(member);
}

// GET: Member/Delete/5
0 references
public async Task<IActionResult> Delete(int? id)...

// POST: Member/Delete/5
[HttpPost, ActionName("Delete")]
[ValidateAntiForgeryToken]
0 references
public async Task<IActionResult> DeleteConfirmed(int id)
{
    var member = await _context.Members.FindAsync(id);
    _context.Members.Remove(member);
    await _context.SaveChangesAsync();
    return RedirectToAction(nameof(Index));
}

1 reference
private bool MemberExists(int id)...
```

1. **Edit(int? id):**
    1. Düzenlemek için bir üyenin ayrıntılarını getirir.
    2. Üye bilgilerini değiştirmek için bir görünüm oluşturur.
2. **Edit(int id, [Bind("Id,Name,Surname,PhoneNumber,PasswordMember,UserName,AdminRol")] Member member):**
    1. Üye güncelleme isteklerini işler.
    2. Değişiklikleri doğrular.
    3. Üyeyi veritabanında günceller.
    4. Üye listesine yönlendirir.
3. **Delete(int id):**
    1. Bir üyeyi silme için bir onay sayfası görüntüler.
4. **DeleteConfirmed(int id):**
    1. Üye silme isteklerini işler.
    2. Üyeyi veritabanından kaldırır.
    3. Üye listesine yönlendirir.

Views

Web uygulamasının kullanıcı tarafından görülen içeriğini oluşturan HTML, CSS ve JavaScript dosyalarının bir koleksiyonudur.
Kısaca, web uygulamalarının kullanıcı arayüzünü oluşturan dosyalardır.

Home/Login.cshtml

- **Bu sayfa, bir üyenin sisteme giriş yapmasını sağlayan bir giriş sayfasıdır.**

İşlevleri:

- **Kullanıcı adı ve şifre girişi için bir form sunar.**
- **Giriş bilgilerini "Login" adlı bir eyleme gönderir.**
- **Üye olmayanlar için kayıt bağlantısı sağlar.**
- **Kütüphane arka planlı görsel bir tasarıma sahiptir.**

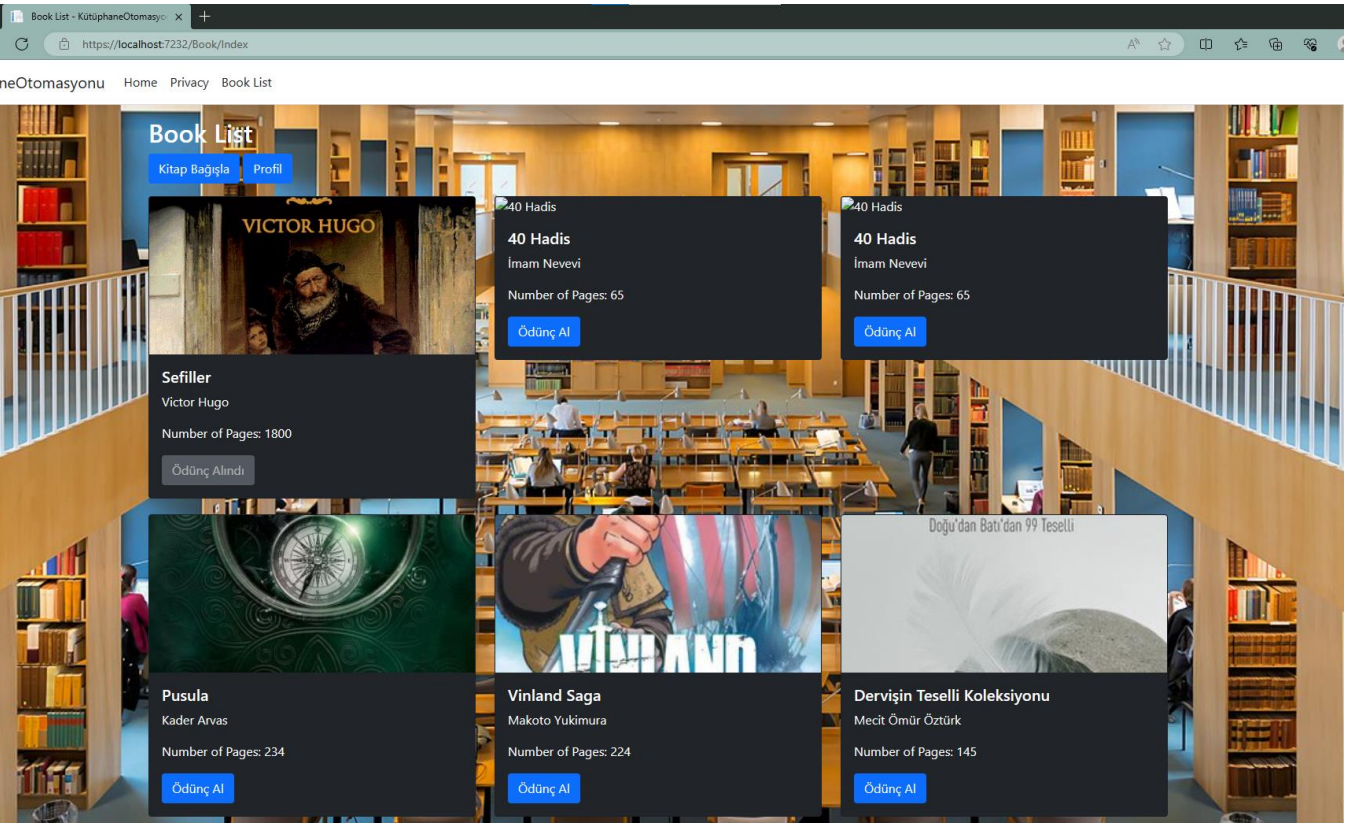
```
@{
    Layout = "_Layout";
}

<!DOCTYPE html>
<html lang="tr">
<head>
    <meta charset="utf-8" />
    <title>Üye Girişi</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" integrity="sha
    <style>
        body {...
        }

        .container {...
        }

        h2 {...
        }

        form {...
        }

        label {...
        }

        button {...
        }
    </style>
</head>
<body>
    <div class="container">
        <h2>Üye Girişi</h2>

        <form asp-action="Login" method="post">
            <div class="form-group">
                <label for="Username">Kullanıcı Adı:</label>
                <input type="text" class="form-control" id="Username" name="Username" required>
            </div>

            <div class="form-group">
                <label for="Password">Şifre:</label>
                <input type="password" class="form-control" id="Password" name="Password" required>
            </div>

            <button type="submit" class="btn btn-primary">Giriş Yap</button>

        <div class="register-link">
            <a href="@Url.Action("Create", "Member")" class="btn btn-success">Üye Ol</a>
        </div>

    </div>

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js" integrity="sha384-mQ93GR6
</body>
</html>
```

Member/Create.cshtml

Bu sayfa kısaca özetlemek gerekirse yeni üye kaydı yapar☺

Book/Index.cshtml

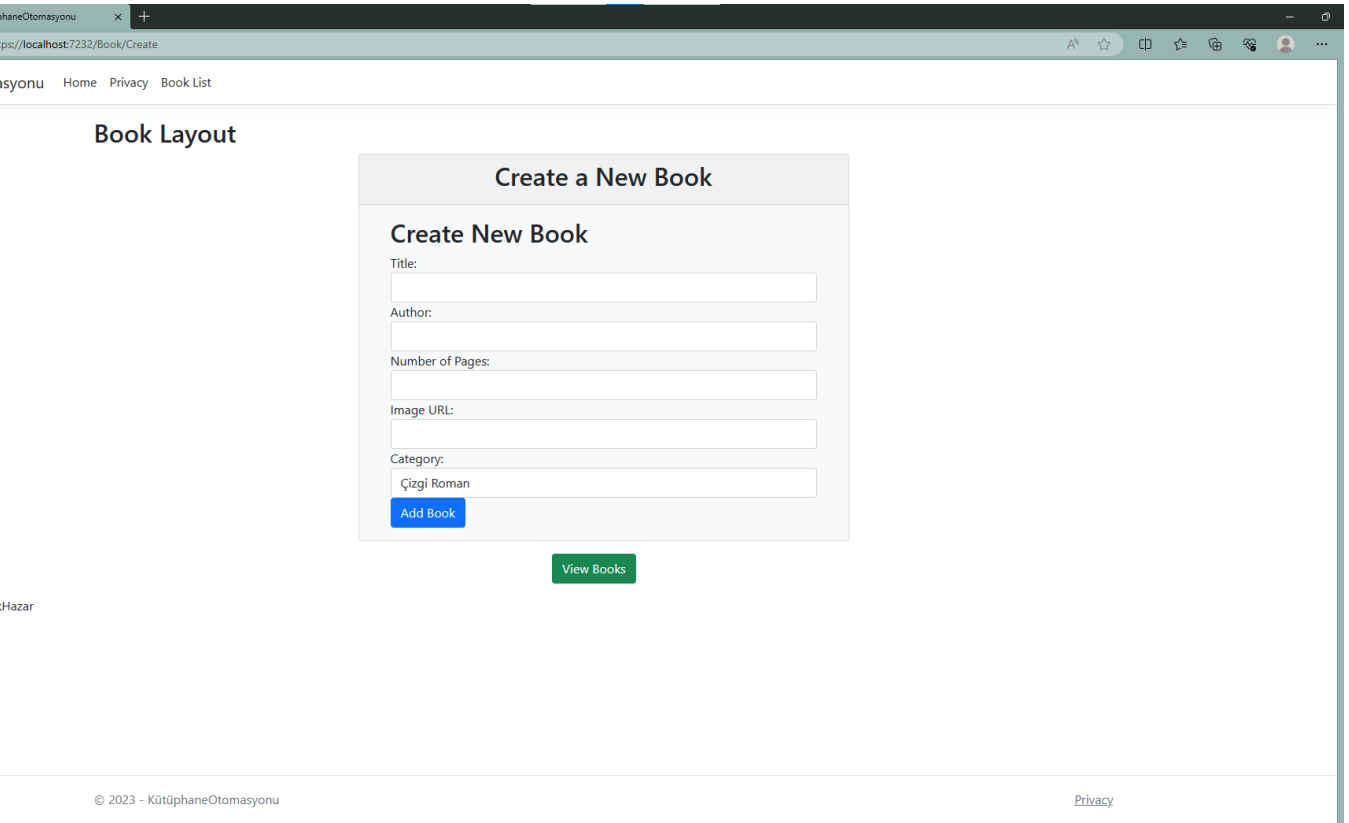
- Bu sayfa giriş yapıldıktan sonra kullanıcıya kitapları göstermek için oluşturulmuştur
- Bu sayfada kullanıcı kitap bağışlayabilir veya profil sayfasına gidebilir




```cshtml
@model List<Book>
@{
    ViewData["Title"] = "Book List";
    var memberId = ViewData["MemberId"]; // Kullanıcı bilgisini al
}
<!DOCTYPE html>
<html lang="tr">
<head>
    <meta charset="utf-8" />
    <title>Book List</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" in
    <style>…</style>
</head>
<body>

    <div class="container">
        <h2>Book List</h2>
        <div >
            <a class="btn btn-primary mb-3" href="@Url.Action("Create", "Book")">Kitap Bağışla</a>
            <a class="btn btn-primary mb-3" href="@Url.Action("Details", "Member", @ViewData["MemberLog"])">
        </div>
        <!-- "Kitap Bağışla" butonu -->

        <div class="row">
            @foreach (var book in Model)
            {
                <div class="col-md-4">
                    <div class="card">
                        <img src="@book.ImageUrl" class="card-img-top" alt="@book.Title">
                        <div class="card-body">
                            <h5 class="card-title">@book.Title</h5>
                            <p class="card-text">@book.Author</p>
                            <p class="card-text">Number of Pages: @book.Numberofpages</p>

                            @if (book.Isborrowed)
                            {
                                <button type="button" class="btn btn-secondary" disabled>Ödünç Alındı</butto
                            }
                            else
                            {
                                <form id="borrowForm" asp-action="Borrow" asp-controller="Book" method="post
                                    <input type="hidden" name="bookId" value="@book.Id" />
                                    <input type="hidden" name="memberId" value="@memberId" />
                                    <button type="button" class="btn btn-primary" onclick="location.href='@U

                                </form>
                            }
                        </div>
                    </div>
                </div>
            }
        </div>

        @if (Model.Count == 0)...
    </div>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js" integrity="sh
</body>
</html>
```

Book/Create.cshtml

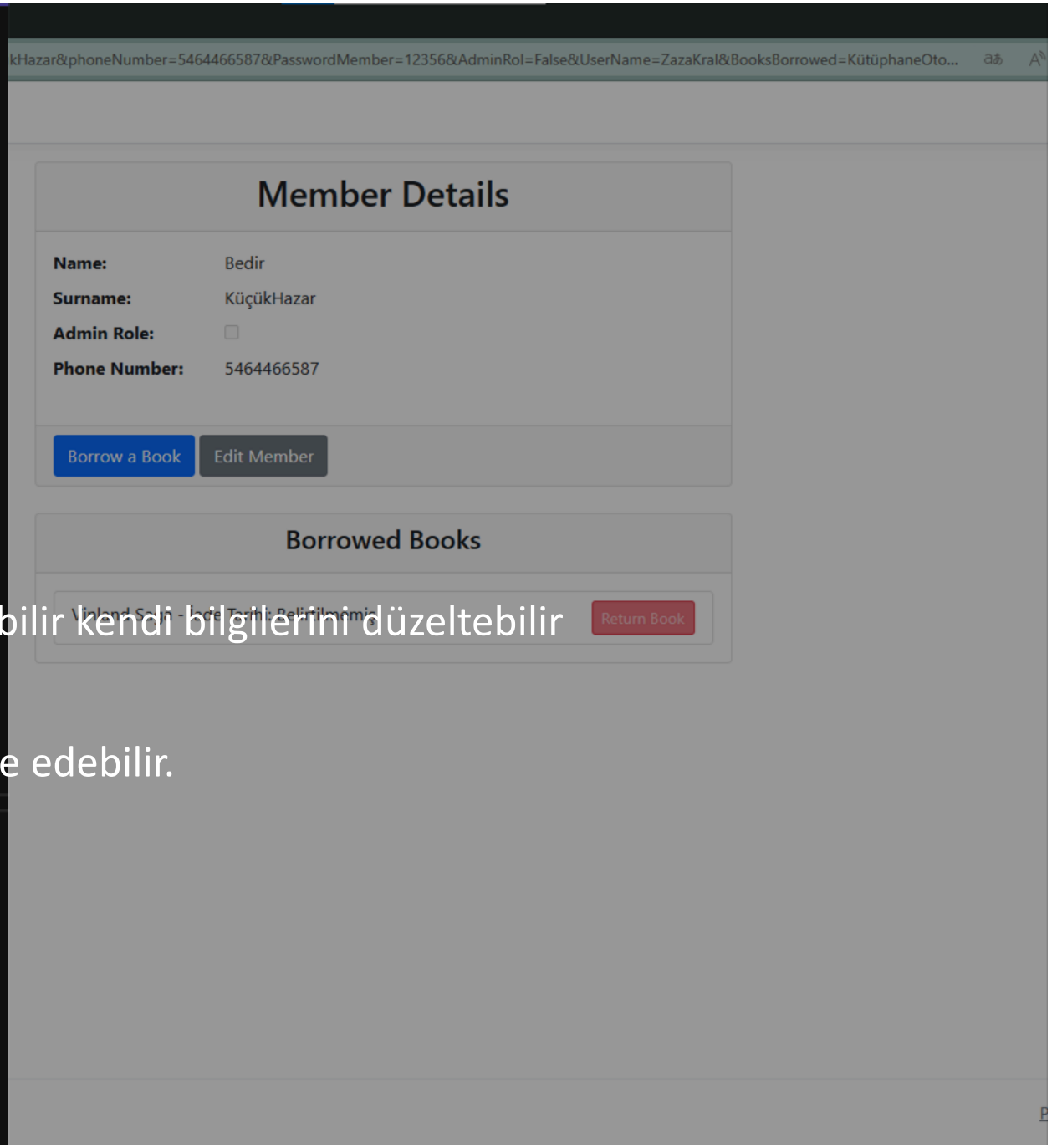
- Bu sayfada giriş yapan kullanıcı yeni bir kitap eklemek ister ise buradan kitap bilgilerini girerek kitap ekleyebilir

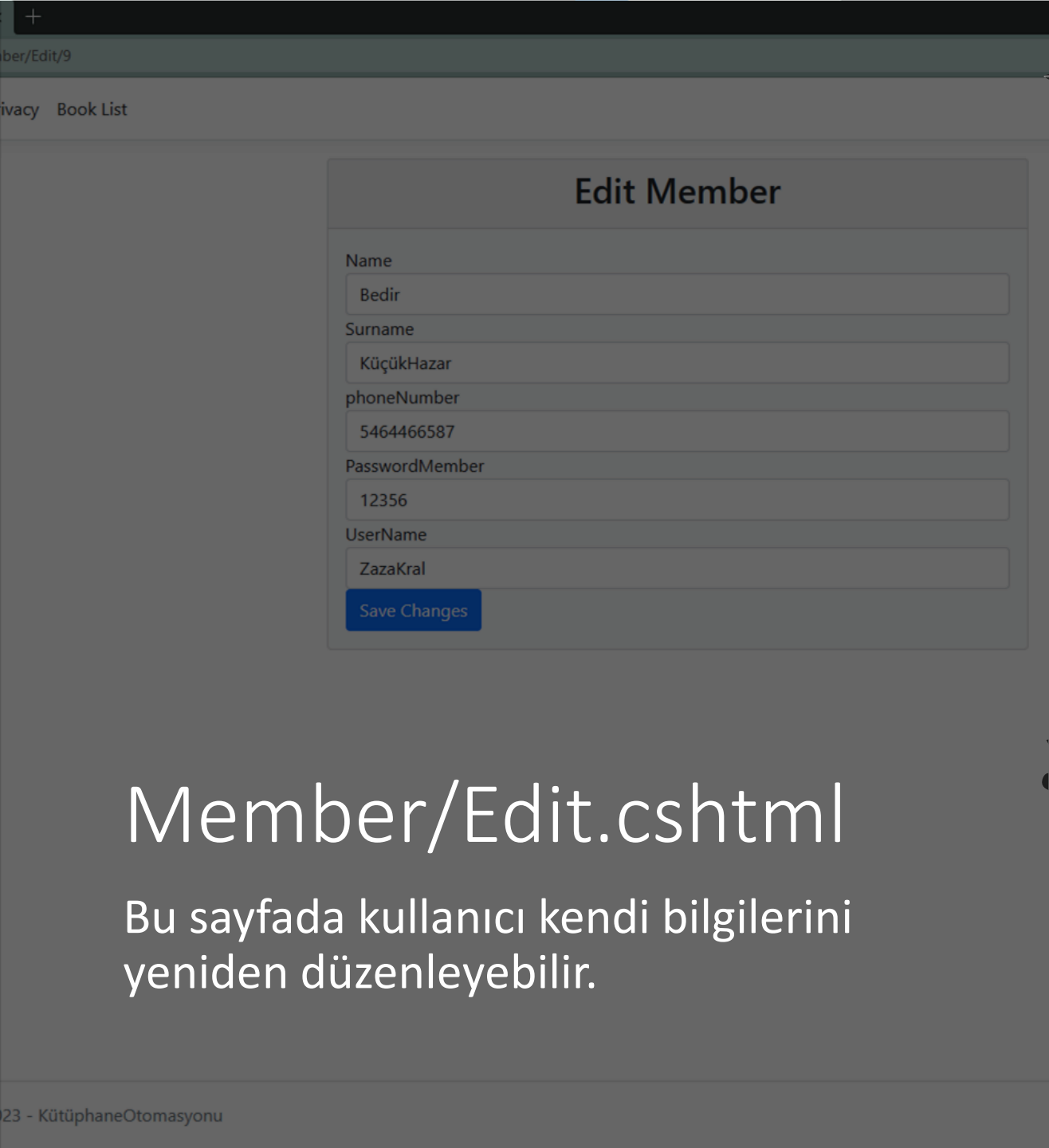



```
@model KütüphaneOtomasyonu.Models.Book

@{
    Layout = "_BookLayout";
}

<!DOCTYPE html>
<html lang="tr">
<head>...</head>
<body>
    <div class="container">
        <div class="row">
            <div class="col-md-6 offset-md-3">
                <div class="card bg-light mb-3">
                    <div class="card-header text-center">
                        <h2>Create a New Book</h2>
                    </div>
                    <div class="card-body">
                        <div class="container">
                            <h2>Create New Book</h2>
                            <form asp-action="Create" method="post">
                                <div class="form-group">
                                    <label for="Title">Title:</label>
                                    <input type="text" class="form-control" id="Title" name="Title" required>
                                </div>
                                <div class="form-group">
                                    <label for="Author">Author:</label>
                                    <input type="text" class="form-control" id="Author" name="Author" required>
                                </div>
                                <div class="form-group">
                                    <label for="Numberofpages">Number of Pages:</label>
                                    <input type="number" class="form-control" id="Numberofpages" name="Numberofpages" req
                                </div>
                                <div class="form-group">
                                    <label for="ImageUrl">Image URL:</label>
                                    <input type="text" class="form-control" id="ImageUrl" name="ImageUrl" required>
                                </div>
                                <div class="form-group">
                                    <label for="CategoryId">Category:</label>
                                    <select class="form-control" id="CategoryId" name="CategoryId" required>
                                        <option value="1">Çizgi Roman</option>
                                        <option value="2">Roman</option>
                                        <option value="3">Biyografi</option>
                                        <option value="4">OtoBiyografi</option>
                                        <option value="5">Hikaye</option>
                                        <option value="6">Makale</option>
                                        <option value="7">Dini Bilgiler</option>
                                        <option value="8">Araştırma</option>
                                        <option value="9">Masal</option>
                                        <option value="10">Şiir</option>
                                        <option value="11">Deneme</option>
                                    </select>
                                </div>
                                <button type="submit" class="btn btn-primary">Add Book</button>
                            </form>
                        </div>
                    </div>
                </div>
            </div>
        </div>
        <div class="row">
            <div class="col-md-12 text-center">
                <a class="btn btn-success" href="@Url.Action("Index", "Book")">View Books</a>
            </div>
        </div>
    </div>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js" integrity="sha384-mQ93GR66B
</body>
</html>
```

Member/Details.cshtml

- Bu sayfada giriş yapan üye kendi bilgilerini görebilir kendi bilgilerini düzeltebilir
- Giriş yapan kullanıcı aldığı kitabı görebilir ve iade edebilir.





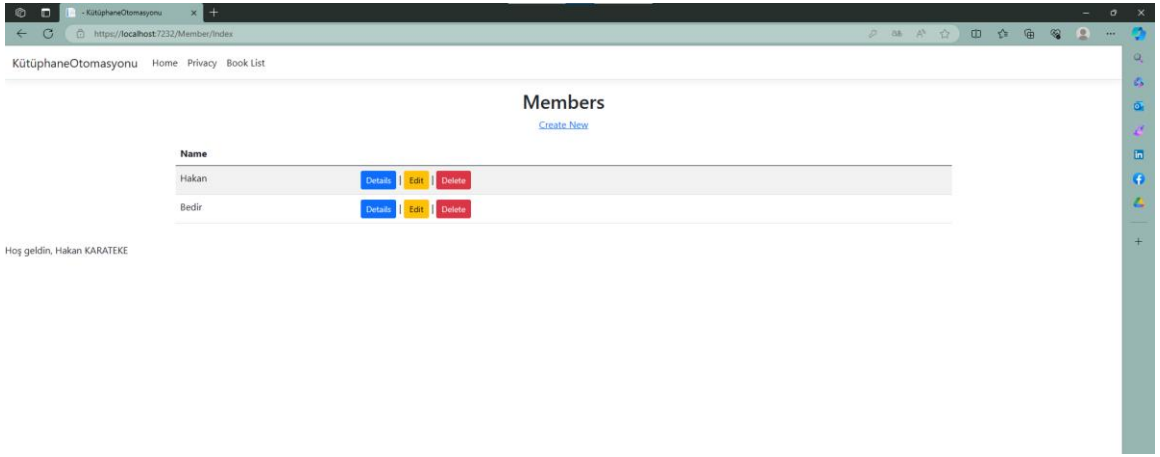
masyonu.Models.Member

```
"container">
class="row">
<div class="col-md-6 offset-md-3">
    <div class="card bg-light mb-3">
        <div class="card-header text-center">
            <h2 class="card-title">Edit Member</h2>
        </div>
        <div class="card-body">
            <form asp-action="Edit" asp-route-id="@Model.Id" method="post">
                <div class="form-group">
                    <label asp-for="Name"></label>
                    <input asp-for="Name" class="form-control" />
                    <span asp-validation-for="Name" class="text-danger"></span>
                </div>
                <div class="form-group">
                    <label asp-for="Surname"></label>
                    <input asp-for="Surname" class="form-control" />
                    <span asp-validation-for="Surname" class="text-danger"></span>
                </div>
                <div class="form-group">
                    <label asp-for="phoneNumber"></label>
                    <input asp-for="phoneNumber" class="form-control" />
                    <span asp-validation-for="phoneNumber" class="text-danger"></span>
                </div>
                <div class="form-group">
                    <label asp-for="PasswordMember"></label>
                    <input asp-for="PasswordMember" class="form-control" />
                    <span asp-validation-for="PasswordMember" class="text-danger"></spa
                </div>
                <div class="form-group">
                    <label asp-for="UserName"></label>
                    <input asp-for="UserName" class="form-control" />
                    <span asp-validation-for="UserName" class="text-danger"></span>
                </div>
                <div class="form-group">
                    <label asp-for="AdminRol">Admin Rol</label>
                    <div class="form-check">
                        <input asp-for="AdminRol" class="form-check-input" />
                        <label class="form-check-label" asp-for="AdminRol">Admin Role</
                    </div>
                    <button type="submit" class="btn btn-primary">Save Changes</button>
                </form>
            </div>
        </div>
```

Member/Edit.cshtml

Bu sayfada kullanıcı kendi bilgilerini yeniden düzenleyebilir.

```cshtml
@model IEnumerable<KütüphaneOtomasyonu.Models.Member>

<div class="container">
    <div class="row">
        <div class="col-md-12">
            <h2 class="text-center">Members</h2>
            <p class="text-center">
                <a asp-action="Create">Create New</a>
            </p>
            <table class="table table-striped table-hover">
                <thead>
                    <tr>
                        <th>Name</th>
                        <th></th>
                    </tr>
                </thead>
                <tbody>
                    @foreach (var item in Model)
                    {
                        <tr>
                            <td>
                                @Html.DisplayFor(modelItem => item.Name)
                            </td>
                            <td>
                                <a asp-action="Details" asp-route-id="@item.Id" class="btn btn-sm btn-primary">Details</a> |
                                <a asp-action="Edit" asp-route-id="@item.Id" class="btn btn-sm btn-warning">Edit</a> |
                                <a asp-action="Delete" asp-route-id="@item.Id" class="btn btn-sm btn-danger">Delete</a>
                            </td>
                        </tr>
                    }
                </tbody>
            </table>
        </div>
    </div>
</div>
```



Member/Index.cshtml

- Bu sayfaya sadece yönetici olan kişilerin erişimi bulunmaktadır.

- Yönetici bu sayfada üyeleri görüntüleyebilir üye silebilir üyenin almış olduğu kitap bilgilerini üyenin detay sayfasına giderek görüntüleyebilir kitabı iade edebilir veya yeni bir üye oluşturabilir.