Laborator 5 - Laborant Andreea Guster

Exercitiul 1 – Validari

[Cerinta] Extindeti codul din laboratorul 4 cu un controller ce implementeaza CRUD pentru tipurile de carti. Adaugati validari simple.

In modelul **Book** aplicati validari pe urmatoarele proprietati:

```
[MinLength(2, ErrorMessage = "Title cannot be less than 2!"),
    MaxLength(200, ErrorMessage = "Title cannot be more than 200!")]
public string Title { get; set; }

[MinLength(3, ErrorMessage = "Author cannot be less than 3!"),
MaxLength(50, ErrorMessage = "Author cannot be more than 50!")]
public string Author { get; set; }

[MinLength(2, ErrorMessage = "Summary cannot be less than 2!"),
MaxLength(5000, ErrorMessage = "Summary cannot be more than 5000!")]
public string Summary { get; set; }
```

In modelul **Publisher** aplicati validari pe urmatoarea proprietate:

```
[MinLength(2, ErrorMessage = "Publisher name cannot be less than 2!"),
MaxLength(30, ErrorMessage = "Publisher name cannot be more than 30!")]
public string Name { get; set; }
```

In modelul **Genre** aplicati validari pe urmatoarea proprietate:

```
[MinLength(3, ErrorMessage = "Genre name cannot be less than 3!"),
MaxLength(20, ErrorMessage = "Genre name cannot be more than 20!")]
public string Name { get; set; }
```

In modelul **ContactInfo** aplicati validari pe urmatoarea proprietate:

```
[RegularExpression(@"^07(\d{8})$", ErrorMessage = "This is not a valid phone number!")]
public string PhoneNumber { get; set; }
```

In modelul **BookType** aplicati validari pentru urmatoarea proprietate:

```
[MinLength(2, ErrorMessage = "Book type name cannot be less than 2!"),
MaxLength(20, ErrorMessage = "Book type name cannot be more than 20!")]
public string Name { get; set; }
```

Daca doriti sa vi se afiseze pe ecran aceste mesaje de eroare, trebui sa va asigurati in view-urile care se ocupa cu modificarea unui obiect existent (**update/edit**) sau crearea unui obiect nou (**new/create**) introduceti urmatoare linie de cod pentru fiecare din proprietatile ce contin validari:

```
@Html.ValidationMessageFor(b => b.Title, "", new { @class = "text-danger" })
```

Exercitiul 1 – CRUD pentru BookType

Creati in directorul **Controllers** fisierul **BookTypeController.cs** si adaugati actiunile **Index**, **New**, **Update**, **Delete**:

```
using lab3_miercuri.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
```

```
using System.Web.Mvc;

namespace lab3_miercuri.Controllers
{
    public class BookTypeController : Controller
    {
        private DbCtx db = new DbCtx();

        public ActionResult Index()
        {
            ViewBag.BookTypes = db.BookTypes.ToList();
            return View();
        }

        public ActionResult New()
        {
            BookType bookType = new BookType();
            return View(bookType);
        }

        [HttpPost]
        public ActionResult New(BookType bookTypeRequest)
        {
            try
            {
                if (ModelState.IsValid)
                {
                    db.BookTypes.Add(bookTypeRequest);
                    db.SaveChanges();
                    return RedirectToAction("Index");
                }
                return View(bookTypeRequest);
            }
            catch (Exception e)
            {
                return View(bookTypeRequest);
            }
        }

        public ActionResult Edit(int? id)
        {
            if (id.HasValue)
            {
                BookType bookType = db.BookTypes.Find(id);

                if (bookType == null)
                {
                    return HttpNotFound("Couldn't find the book type with id " + id.ToString() + "!");
                }
                return View(bookType);
            }
            return HttpNotFound("Couldn't find the book type with id " + id.ToString() + "!");
        }

        [HttpPut]
        public ActionResult Edit(int id, BookType bookTypeRequestor)
        {
            try
            {
                if (ModelState.IsValid)
                {
                    BookType bookType = db.BookTypes.Find(id);
                    if (TryUpdateModel(bookType))
                    {
                        bookType.Name = bookTypeRequestor.Name;
                        db.SaveChanges();
                    }
                    return RedirectToAction("Index");
                }
```

```
                return View(bookTypeRequestor);
            }
            catch(Exception e)
            {
                return View(bookTypeRequestor);
            }
        }

        [HttpDelete]
        public ActionResult Delete(int? id)
        {
            if (id.HasValue)
            {
                BookType bookType = db.BookTypes.Find(id);
                if(bookType != null)
                {
                    db.BookTypes.Remove(bookType);
                    db.SaveChanges();
                    return RedirectToAction("Index");
                }
                return HttpNotFound("Couldn't find the book type with id " + id.ToString() + "!");
            }
            return HttpNotFound("Book type id parameter is missing!");
        }
    }
}
```

In **Views/BookType** creati view-ul **Index.cshtml**:

```
@{
    ViewBag.Title = "Book Types";
}

<h2>@ViewBag.Title</h2>

<table class="table">
    <thead>
        <th>Name</th>
        <th>Update/Remove</th>
    </thead>
    <tbody>
        @foreach (var bookType in ViewBag.BookTypes)
        {
            <tr>
                <td>@bookType.Name</td>
                <td>
                    @using (Html.BeginForm(actionName: "Edit", controllerName: "BookType", method:
FormMethod.Get,
                        routeValues: new { id = bookType.BookTypeId }))
                    {
                        <button type="submit" class="btn btn-link col-lg-3">Update</button>
                    }

                    @using (Html.BeginForm(actionName: "Delete", controllerName: "BookType", method:
FormMethod.Post,
                        routeValues: new { id = bookType.BookTypeId }))
                    {
                        @Html.HttpMethodOverride(HttpVerbs.Delete)
                        <button type="submit" class="btn btn-link col-lg-3">Remove</button>
                    }
                </td>
            </tr>
        }
    </tbody>
</table>
<br />
@Html.ActionLink("Add book type", "New")
```

In **Views/BookType** creati view-ul **New.cshtml**:

```
@model lab3_miercuri.Models.BookType
@{
    ViewBag.Title = "Create a book type";
}

<h2>@ViewBag.Title</h2>

@using (Html.BeginForm(actionName: "New", controllerName: "BookType", method: FormMethod.Post))
{
    @Html.LabelFor(bt => bt.Name, "Name:")
    <br />
    @Html.TextBoxFor(bt => bt.Name, null, new { placeholder = "Type in the book type's name", @class =
"form-control" })
    @Html.ValidationMessageFor(bt => bt.Name, "", new { @class = "text-danger" })
    <br /><br />

    <button class="btn btn-primary" type="submit">Create</button>
}
```

In **Views/BookType** creati view-ul **Edit.cshtml**:

```
@model lab3_miercuri.Models.BookType
@{
    ViewBag.Title = "Update book type";
}

<h2>@ViewBag.Title</h2>

@using (Html.BeginForm(actionName: "Edit", controllerName: "BookType", method: FormMethod.Post))
{
    @Html.HttpMethodOverride(HttpVerbs.Put)
    @Html.HiddenFor(b => b.BookTypeId)


    @Html.LabelFor(bt => bt.Name, "Name:")
    <br />
    @Html.EditorFor(bt => bt.Name, new { htmlAttributes = new { @class = "form-control" } })
    @Html.ValidationMessageFor(bt => bt.Name, "", new { @class = "text-danger" })
    <br /><br />

    <button class="btn btn-primary" type="submit">Update</button>
}
```

Exercitiul 2 – Validare nr prim

[Cerinta] Creati un validator personalizat ce testeaza daca un numar e prim.

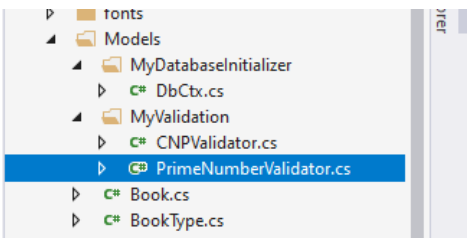
In clasa **Book** din directorul **Models** adauagti urmatoarea proprietate:

```
[PrimeNumberValidator]
public int NoOfPages { get; set; }
```

Tot in aceasta clasa trebuie sa **importati validarea** pe care o vom crea, astfel:

```
using lab3_miercuri.Models.MyValidation;
```

Vom crea un **director** nou in interiorul directorului **Models** – **/Models/MyValidation/** in care vom crea clasa **PrimeNumberValidator**:



```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;

namespace lab3_miercuri.Models
{
    public class PrimeNumberValidator : ValidationAttribute
    {
        protected override ValidationResult IsValid(object value, ValidationContext validationContext)
        {
            var book = (Book)validationContext.ObjectInstance;
            int pages = book.NoOfPages;
            bool cond = true;

            for (int i = 2; i <= Math.Sqrt(pages); i++)
            {
                if (pages % i == 0)
                {
                    cond = false;
                    break;
                }
            }
            return cond ? ValidationResult.Success : new ValidationResult("This is not a prime number!");
        }
    }
}
```

In **Views/Book/New.cshtml** dupa ce afisam textbox-ul pt numele autorului, scrieti urmatoarea secventa de cod:

```
    @Html.Label("NoOfPages", " Number of pages:")
    <br />
    @Html.TextBoxFor(b => b.NoOfPages, null, new { placeholder = "Type in the book's number of pages",
@class = "form-control" })
    @Html.ValidationMessageFor(b => b.NoOfPages, "", new { @class = "text-danger" })
    <br />
```

In **Views/Book/Edit.cshtml** dupa ce afisam editor-ul pt numele autorului, scrieti urmatoarea secventa de cod:

```
    @Html.Label("NoOfPages", "Number of pages:")
    <br />
    @Html.EditorFor(b => b.NoOfPages,
        new { htmlAttributes = new { @class = "form-control", @type="number" } })
    @Html.ValidationMessageFor(b => b.NoOfPages, "", new { @class = "text-danger" })
    <br />
```

In **Controllers/BookController** in actiunea **Edit** de tipul PUT adaugati linia cu rosu:

```
                if (TryUpdateModel(book))
                {
                    book.Title = bookRequest.Title;
                    book.Author = bookRequest.Author;
                    book.Summary = bookRequest.Summary;
                    book.NoOfPages = bookRequest.NoOfPages;
                    db.SaveChanges();
                }
```

Exercitiul 3 – Validare CNP

In clasa **Initp** in interiorul metodei **Seed** adaugati:

```
    using System;
using System.Collections.Generic;
```

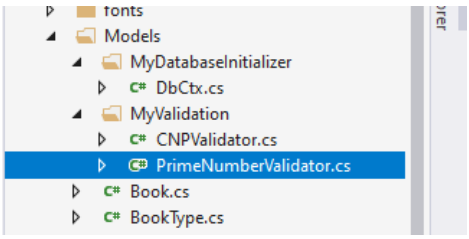
```
using System.Data.Entity;
using System.Linq;
using System.Web;

namespace lab3_miercuri.Models.MyDatabaseInitializer
{

    public class DbCtx : DbContext
    {
        public DbCtx() : base("DbConnectionString")
        {
            Database.SetInitializer<DbCtx>(new Initp());
            //Database.SetInitializer<DbCtx>(new CreateDatabaseIfNotExists<DbCtx>());
            //Database.SetInitializer<DbCtx>(new DropCreateDatabaseIfModelChanges<DbCtx>());
            //Database.SetInitializer<DbCtx>(new DropCreateDatabaseAlways<DbCtx>());
        }


        public DbSet<Book> Books { get; set; }
        public DbSet<Publisher> Publishers { get; set; }
        public DbSet<Genre> Genres { get; set; }
        public DbSet<ContactInfo> ContactsInfo { get; set; }
        public DbSet<BookType> BookTypes { get; set; }
        public DbSet<Region> Regions { get; set; }
    }

    public class Initp : DropCreateDatabaseAlways<DbCtx>
    {
        protected override void Seed(DbCtx ctx)
        {
            BookType cover1 = new BookType { BookTypeId = 67, Name = "Hardcover" };
            BookType cover2 = new BookType { BookTypeId = 68, Name = "Paperback" };

            Region region1 = new Region { RegionId = 1, Name = "Alba" };
            Region region2 = new Region { RegionId = 2, Name = "Arad" };
            Region region3 = new Region { RegionId = 3, Name = "Argeș" };
            Region region4 = new Region { RegionId = 4, Name = "Bacău" };

            ctx.BookTypes.Add(cover1);
            ctx.BookTypes.Add(cover2);

            ctx.Regions.Add(region1);
            ctx.Regions.Add(region2);
            ctx.Regions.Add(region3);
            ctx.Regions.Add(region4);
            ctx.Regions.Add(new Region { RegionId = 5, Name = "Bihor" });
            ctx.Regions.Add(new Region { RegionId = 6, Name = "Bistrița-Năsăud" });
            ctx.Regions.Add(new Region { RegionId = 7, Name = "Botoșani" });
            ctx.Regions.Add(new Region { RegionId = 8, Name = "Brașov" });
            ctx.Regions.Add(new Region { RegionId = 9, Name = "Brăila" });
            ctx.Regions.Add(new Region { RegionId = 10, Name = "Buzău" });
            ctx.Regions.Add(new Region { RegionId = 11, Name = "Caraș-Severin" });
            ctx.Regions.Add(new Region { RegionId = 12, Name = "Cluj" });
            ctx.Regions.Add(new Region { RegionId = 13, Name = "Constanța" });
            ctx.Regions.Add(new Region { RegionId = 14, Name = "Covasna" });
            ctx.Regions.Add(new Region { RegionId = 15, Name = "Dâmbovița" });
            ctx.Regions.Add(new Region { RegionId = 16, Name = "Dolj" });
            ctx.Regions.Add(new Region { RegionId = 17, Name = "Galați" });
            ctx.Regions.Add(new Region { RegionId = 18, Name = "Gorj" });
            ctx.Regions.Add(new Region { RegionId = 19, Name = "Harghita" });
            ctx.Regions.Add(new Region { RegionId = 20, Name = "Hunedoara" });
            ctx.Regions.Add(new Region { RegionId = 21, Name = "Ialomița" });
            ctx.Regions.Add(new Region { RegionId = 22, Name = "Iași" });
            ctx.Regions.Add(new Region { RegionId = 23, Name = "Ilfov" });
            ctx.Regions.Add(new Region { RegionId = 24, Name = "Maramureș" });
            ctx.Regions.Add(new Region { RegionId = 25, Name = "Mehedinți" });
            ctx.Regions.Add(new Region { RegionId = 26, Name = "Mureș" });
            ctx.Regions.Add(new Region { RegionId = 27, Name = "Neamț" });
            ctx.Regions.Add(new Region { RegionId = 28, Name = "Olt" });
```

```
ctx.Regions.Add(new Region { RegionId = 29, Name = "Prahova" });
ctx.Regions.Add(new Region { RegionId = 30, Name = "Satu Mare" });
ctx.Regions.Add(new Region { RegionId = 31, Name = "Sălaj" });
ctx.Regions.Add(new Region { RegionId = 32, Name = "Sibiu" });
ctx.Regions.Add(new Region { RegionId = 33, Name = "Suceava" });
ctx.Regions.Add(new Region { RegionId = 34, Name = "Teleorman" });
ctx.Regions.Add(new Region { RegionId = 35, Name = "Timiș" });
ctx.Regions.Add(new Region { RegionId = 36, Name = "Tulcea" });
ctx.Regions.Add(new Region { RegionId = 37, Name = "Vaslui" });
ctx.Regions.Add(new Region { RegionId = 38, Name = "Vâlcea" });
ctx.Regions.Add(new Region { RegionId = 39, Name = "Vrancea" });
ctx.Regions.Add(new Region { RegionId = 40, Name = "București" });
ctx.Regions.Add(new Region { RegionId = 41, Name = "București-Sector 1" });
ctx.Regions.Add(new Region { RegionId = 42, Name = "București-Sector-2" });
ctx.Regions.Add(new Region { RegionId = 43, Name = "București-Sector-3" });
ctx.Regions.Add(new Region { RegionId = 44, Name = "București-Sector-4" });
ctx.Regions.Add(new Region { RegionId = 45, Name = "București-Sector 5" });
ctx.Regions.Add(new Region { RegionId = 46, Name = "București - Sector 6" });
ctx.Regions.Add(new Region { RegionId = 51, Name = "Călărași" });
ctx.Regions.Add(new Region { RegionId = 52, Name = "Giurgiu" });

ContactInfo contact1 = new ContactInfo
{
    PhoneNumber = "0712345678",
    BirthDay = "16",
    BirthMonth = "03",
    BirthYear = 1991,
    CNP = "2910316014007",
    GenderType = Gender.Female,
    Resident = false,
    RegionId = region1.RegionId
};

ContactInfo contact2 = new ContactInfo
{
    PhoneNumber = "0713345878",
    BirthDay = "07",
    BirthMonth = "04",
    BirthYear = 2002,
    CNP = "6020407023976",
    GenderType = Gender.Female,
    Resident = false,
    RegionId = region2.RegionId
};

ContactInfo contact3 = new ContactInfo
{
    PhoneNumber = "0711345678",
    BirthDay = "30",
    BirthMonth = "10",
    BirthYear = 2002,
    CNP = "5021030031736",
    GenderType = Gender.Male,
    Resident = false,
    RegionId = region3.RegionId
};

ContactInfo contact4 = new ContactInfo
{
    PhoneNumber = "0712665679",
    BirthDay = "13",
    BirthMonth = "05",
    BirthYear = 1986,
    CNP = "2860513040701",
    GenderType = Gender.Female,
    Resident = false,
    RegionId = region4.RegionId
};
```

```
            ctx.ContactsInfo.Add(contact1);
            ctx.ContactsInfo.Add(contact2);
            ctx.ContactsInfo.Add(contact3);
            ctx.ContactsInfo.Add(contact4);

            ctx.Books.Add(new Book
            {
                Title = "The Atomic Times",
                Author = "Michael Harris",
                Publisher = new Publisher { Name = "HarperCollins", ContactInfo = contact1 },
                BookType = cover1,
                Genres = new List<Genre> {
                    new Genre { Name = "Horror"}
                }
            });
            ctx.Books.Add(new Book
            {
                Title = "In Defense of Elitism",
                Author = "Joel Stein",
                Publisher = new Publisher { Name = "Macmillan Publishers", ContactInfo = contact2 },
                BookType = cover1,
                Genres = new List<Genre> {
                    new Genre { Name = "Humor"}
                }
            });
            ctx.Books.Add(new Book
            {
                Title = "The Canterbury Tales",
                Author = "Geoffrey Chaucer",
                Summary = "At the Tabard Inn, a tavern in Southwark, near London, the narrator joins a
company of twenty-nine pilgrims. The pilgrims, like the narrator, are traveling to the shrine of the
martyr Saint Thomas Becket in Canterbury. The narrator gives a descriptive account of twenty-seven of
these pilgrims, including a Knight, Squire, Yeoman, Prioress, Monk, Friar, Merchant, Clerk, Man of Law,
Franklin, Haberdasher, Carpenter, Weaver, Dyer, Tapestry-Weaver, Cook, Shipman, Physician, Wife, Parson,
Plowman, Miller, Manciple, Reeve, Summoner, Pardoner, and Host. (He does not describe the Second Nun or
the Nun’s Priest, although both characters appear later in the book.) The Host, whose name, we find out in
the Prologue to the Cook’s Tale, is Harry Bailey, suggests that the group ride together and entertain one
another with stories. He decides that each pilgrim will tell two stories on the way to Canterbury and two
on the way back. Whomever he judges to be the best storyteller will receive a meal at Bailey’s tavern,
courtesy of the other pilgrims. The pilgrims draw lots and determine that the Knight will tell the first
tale.",
                Publisher = new Publisher { Name = "Scholastic", ContactInfo = contact3 },
                BookType = cover2,
                Genres = new List<Genre> {
                    new Genre { Name = "Satire"},
                    new Genre { Name = "Fabilau"},
                    new Genre { Name = "Allegory"},
                    new Genre { Name = "Burlesque"}
                }
            });
            ctx.Books.Add(new Book
            {
                Title = "Python Crash Course, 2nd Edition: A Hands-On, Project-Based Introduction to
Programming",
                Author = "Eric Matthers",
                Publisher = new Publisher { Name = "Schol", ContactInfo = contact4 },
                BookType = cover2,
                Genres = new List<Genre> {
                    new Genre { Name = "Programming"}
                }
            });

            ctx.SaveChanges();
            base.Seed(ctx);
        }
    }
}
```

In directorul **/Models/MyValidation/** in care vom crea clasa **CNPValidator**:

```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using lab3_miercuri.Models.MyValidation;

namespace lab3_miercuri.Models
{
    [Table("Contacts")]
    public class ContactInfo
    {
        public int ContactInfoId { get; set; }

        [RegularExpression(@"^07(\d{8})$", ErrorMessage = "This is not a valid phone number!")]
        public string PhoneNumber { get; set; }

        [CNPValidator]
        public string CNP { get; set; }

        [Required, RegularExpression(@"^[1-9](\d{3})$", ErrorMessage = "This is not a valid year!")]
        public int BirthYear { get; set; }

        [Required, RegularExpression(@"^(0[1-9])|(1[012])$", ErrorMessage = "This is not a valid month!")]
        public string BirthMonth { get; set; }

        [Required, RegularExpression(@"^((0[1-9])|([12]\d)|(3[01]))$", ErrorMessage = "This is not a valid
day number!")]
        public string BirthDay { get; set; }

        [Required]
        public Gender GenderType { get; set; }

        public bool Resident { get; set; }

        //one-to-many
        public int RegionId { get; set; }
        public virtual Region Region { get; set; }

        // one-to-one relationship
        public virtual Publisher Publisher { get; set; }
    }

    public enum Gender
    {
        Male,
        Female
    }
}
```



In directorul **Models** creati clasa **Region**:

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Web;

namespace lab3_miercuri.Models
{
    public class Region
    {
        public int RegionId { get; set; }

        [MinLength(2, ErrorMessage = "Region name cannot be less than 2!"),
        MaxLength(50, ErrorMessage = "Region name cannot be more than 50!")]
        public string Name { get; set; }

        // many-to-one
```

```
        public virtual ICollection<ContactInfo> ContactsInfo { get; set; }
    }
}
```

In **/Models/MyValidator** adaugati clasa **CNPValidator**:

```
using lab3_miercuri.Models.MyDatabaseInitializer;
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Text.RegularExpressions;
using System.Web;

namespace lab3_miercuri.Models.MyValidation
{
    public class CNPValidator : ValidationAttribute
    {
        private bool BeUnique(string cnp)
        {
            DbCtx db = new DbCtx();
            return db.ContactsInfo.FirstOrDefault(ci => ci.CNP == cnp) == null;
        }

        protected override ValidationResult IsValid(object value, ValidationContext validationContext)
        {
            ContactInfo contact = (ContactInfo)validationContext.ObjectInstance;

            string cnp = contact.CNP;

            if (cnp == null)
                return new ValidationResult("CNP field is required!");

            Regex regex = new Regex(@"^([1-9]\d+)$");
            if (!regex.IsMatch(cnp))
                return new ValidationResult("CNP must contain only digits!");

            if (cnp.Length != 13)
                return new ValidationResult("CNP must contain 13 digits!");

            if (!BeUnique(cnp))
                return new ValidationResult("CNP must be unique!");

            char s = cnp.ElementAt(0);
            string aa = cnp.Substring(1, 2);
            string ll = cnp.Substring(3, 2);
            string zz = cnp.Substring(5, 2);
            string jj = cnp.Substring(7, 2);
            string nnn = cnp.Substring(9, 3);
            char c = cnp.ElementAt(12);

            switch (s)
            {
                case '1':
                    if (!(contact.GenderType.Equals(Gender.Male) && contact.BirthYear >= 1900 &&
contact.BirthYear <= 1999))
                        return new ValidationResult("S part is not valid!");
                    break;
                case '2':
                    if (!(contact.GenderType.Equals(Gender.Female) && contact.BirthYear >= 1900 &&
contact.BirthYear <= 1999))
                        return new ValidationResult("S part is not valid!");
                    break;
                case '3':
                    if (!(contact.GenderType.Equals(Gender.Male) && contact.BirthYear >= 1800 &&
contact.BirthYear <= 1899))
                        return new ValidationResult("S part is not valid!");
                    break;
```

```
                case '4':
                    if (!(contact.GenderType.Equals(Gender.Female) && contact.BirthYear >= 1800 &&
contact.BirthYear <= 1899))
                        return new ValidationResult("S part is not valid!");
                    break;
                case '5':
                    if (!(contact.GenderType.Equals(Gender.Male) && contact.BirthYear >= 2000 &&
contact.BirthYear <= 2099))
                        return new ValidationResult("S part is not valid!");
                    break;
                case '6':
                    if (!(contact.GenderType.Equals(Gender.Female) && contact.BirthYear >= 2000 &&
contact.BirthYear <= 2099))
                        return new ValidationResult("S part is not valid!");
                    break;
                case '7':
                    if (!(contact.GenderType.Equals(Gender.Male) && contact.Resident.Equals(true)))
                        return new ValidationResult("S part is not valid!");
                    break;
                case '8':
                    if (!(contact.GenderType.Equals(Gender.Female) && contact.Resident.Equals(true)))
                        return new ValidationResult("S part is not valid!");
                    break;
                default: return new ValidationResult("S part is not valid!");
            }

            string lastTwoDigits = contact.BirthYear.ToString().Substring(2, 2);
            if (!aa.Equals(lastTwoDigits))
                return new ValidationResult("AA part is not valid!");

            if (!ll.Equals(contact.BirthMonth))
                return new ValidationResult("LL part is not valid!");

            if (!zz.Equals(contact.BirthDay))
                return new ValidationResult("ZZ part is not valid!");

               int regionNo = contact.RegionId;
               string region;

               if (regionNo < 10)
                   region = "0" + regionNo.ToString();
               else
                   region = regionNo.ToString();

              if (!jj.Equals(region))
                   return new ValidationResult("JJ part is not valid!");

             regex = new Regex(@"^((00[1-9])|(0[1-9]\d)|([1-9]\d{2}))$");
             if (!regex.IsMatch(nnn))
                 return new ValidationResult("NNN part is not valid!");

           // validam componenta C
           int rez = (s - '0') * 2;
           rez += (aa.ElementAt(0) - '0') * 7 + (aa.ElementAt(1) - '0') * 9;
           rez += (ll.ElementAt(0) - '0') * 1 + (ll.ElementAt(1) - '0') * 4;
           rez += (zz.ElementAt(0) - '0') * 6 + (zz.ElementAt(1) - '0') * 3;
           rez += (jj.ElementAt(0) - '0') * 5 + (jj.ElementAt(1) - '0') * 8;
           rez += (nnn.ElementAt(0) - '0') * 2 + (nnn.ElementAt(1) - '0') * 7 + (nnn.ElementAt(2) - '0') *
9;
           rez %= 11;
           if (rez == 10)
           {
               if (!c.Equals('1'))
                   return new ValidationResult("C part is not valid!");
           }
           else
           {
               if ((c - '0') != rez)
                   return new ValidationResult("C part is not valid!");
```

```
            }
             return ValidationResult.Success;
        }
    }
}
```

In **ContactInfoController** adaugati operatia de crearea a unui obiect nou:

```
using lab3_miercuri.Models;
using lab3_miercuri.Models.MyDatabaseInitializer;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace lab3_miercuri.Controllers
{
    public class ContactInfoController : Controller
    {
        private DbCtx db = new DbCtx();

        [HttpGet]
        public ActionResult New()
        {
            ContactInfo contact = new ContactInfo();
            ViewBag.RegionList = GetAllRegions();
            ViewBag.GenderList = GetAllGenderTypes();
            return View(contact);
        }

        [HttpPost]
        public ActionResult New(ContactInfo contactRequest)
        {
            ViewBag.RegionList = GetAllRegions();
            ViewBag.GenderList = GetAllGenderTypes();
            try
            {
                if (ModelState.IsValid)
                {
                    db.ContactsInfo.Add(contactRequest);
                    db.SaveChanges();
                    return RedirectToAction("Index", "Book");
                }
                return View(contactRequest);
            }
            catch (Exception e)
            {
                return View(contactRequest);
            }
        }



        [NonAction]
        public IEnumerable<SelectListItem> GetAllRegions()
        {
            var selectList = new List<SelectListItem>();
            foreach (var region in db.Regions.ToList())
            {
                selectList.Add(new SelectListItem
                {
                    Value = region.RegionId.ToString(),
                    Text = region.Name
                });
            }
            return selectList;
        }
```

```
        [NonAction]
        public IEnumerable<SelectListItem> GetAllGenderTypes()
        {
            var selectList = new List<SelectListItem>();

            selectList.Add(new SelectListItem
            {
                Value = Gender.Male.ToString(),
                Text = "Male"
            });

            selectList.Add(new SelectListItem
            {
                Value = Gender.Female.ToString(),
                Text = "Female"
            });

            return selectList;
        }
    }
}
```

In **Views/ContactInfo** creati view-ul **New.cshtml**:

```
@model lab3_miercuri.Models.ContactInfo
@{
    ViewBag.Title = "Create a contact";
}

<h2>@ViewBag.Title</h2>
<br />

@using (Html.BeginForm(actionName: "New", controllerName: "ContactInfo", method: FormMethod.Post))
{
    @Html.LabelFor(c => c.PhoneNumber, "Phone number:")
    <br />
    @Html.TextBoxFor(c => c.PhoneNumber, null, new { placeholder = "Type in the phone number", @class =
"form-control" })
    @Html.ValidationMessageFor(c => c.PhoneNumber, "", new { @class = "text-danger" })
    <br />
    <br />

    @Html.LabelFor(c => c.CNP, "CNP:")
    <br />
    @Html.TextBoxFor(c => c.CNP, null, new { placeholder = "Type in the CNP", @class = "form-control" })
    @Html.ValidationMessageFor(c => c.CNP, "", new { @class = "text-danger" })
    <br />
    <br />

    <label>Birth date:</label>
    <br />
    @Html.TextBoxFor(c => c.BirthDay, null, new { placeholder = "Day", @class = "form-control", @style =
"margin-bottom:3px" })
    @Html.ValidationMessageFor(c => c.BirthDay, "", new { @class = "text-danger" })

    @Html.TextBoxFor(c => c.BirthMonth, null, new { placeholder = "Month", @class = "form-control", @style
= "margin-bottom:3px" })
    @Html.ValidationMessageFor(c => c.BirthMonth, "", new { @class = "text-danger" })

    @Html.TextBoxFor(c => c.BirthYear, null, new { placeholder = "Year", @class = "form-control", @style =
"margin-bottom:3px", @type="number" })
    @Html.ValidationMessageFor(c => c.BirthYear, "", new { @class = "text-danger" })
    <br />
    <br />

    @Html.Label("Resident", "Resident:")
    @Html.CheckBoxFor(c => c.Resident, new { @style = "margin-left:10px" })
```

```
    <br />
    <br />

    @Html.Label("GenderType", "Gender:")
    <br />
    @Html.DropDownListFor(c => c.GenderType, new SelectList(ViewBag.GenderList, "Value", "Text"), "Choose
a gender", new { @class = "form-control" })
    @Html.ValidationMessageFor(c => c.GenderType, "", new { @class = "text-danger" })
    <br />
    <br />

    @Html.Label("Regions", "Region:")
    <br />
    @Html.DropDownListFor(c => c.RegionId, new SelectList(ViewBag.RegionList, "Value", "Text"), "Choose a
region", new { @class = "form-control" })
    @Html.ValidationMessageFor(c => c.RegionId, "", new { @class = "text-danger" })
    <br />
    <br />

    <button class="btn btn-primary" type="submit">Create</button>
}
```

Many-To-Many exemplu cu o lista de checkbox-uri

Am adugat aici un exemplu pentru relatia many-to-many intre Book si Genre.

Creare unei carti

Functionalitate: Atunci cand cream o carte noua vrem sa ne apara in interiorul formularului o lista de checkbox-uri cu genurile existente in baza de date.

Pasul 1: Vom crea modelul **CheckBoxViewModel.cs** ce va contine 3 campuri:

- Name = denumire genului literar
- Id = valoarea cheii primare din tabelul Genre
- Checked = ce va lua valoarea **true** daca checkbox-ul a fost bifat
  sau **false** daca a ramas neselectat.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace lab3_miercuri.Models
{
    public class CheckBoxViewModel
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public bool Checked { get; set; }
    }
}
```

Pasul 2: In modelul **Book.cs** vom adauga o lista de checkbox-uri. Fiecare checkbox va reprezenta un gen literar din tabelul Genre al bazei de date.

```
        // checkboxes list
        [NotMapped]
        public List<CheckBoxViewModel> GenresList { get; set; }
```

Pasul 3: In BookController.cs vom adauga o metoda care ne returneaza lista de checkbox-uri cu toate genurile existente in BD.

```
[NonAction]
public List<CheckBoxViewModel> GetAllGenres()
{
    var checkboxList = new List<CheckBoxViewModel>();
    foreach (var genre in db.Genres.ToList())
    {
        checkboxList.Add(new CheckBoxViewModel
        {
            Id = genre.GenreId,
            Name = genre.Name,
            Checked = false
        });
    }
    return checkboxList;
}
```

Pasul 4: Vom modifica metodele **New** de tipul GET si POST din **BookController.cs**.

```
[HttpGet]
public ActionResult New()
{
    Book book = new Book();
    book.BookTypeList = GetAllBookTypes();
    book.PublisherList = GetAllPublishers();
    book.GenresList = GetAllGenres();
    book.Genres = new List<Genre>();
    return View(book);
}

[HttpPost]
public ActionResult New(Book bookRequest)
{
    bookRequest.BookTypeList = GetAllBookTypes();
    bookRequest.PublisherList = GetAllPublishers();

    // memoram intr-o lista doar genurile care au fost selectate
    var selectedGenres = bookRequest.GenresList.Where(b => b.Checked).ToList();
    try
    {
        if (ModelState.IsValid)
        {
            bookRequest.Genres = new List<Genre>();
            for(int i = 0; i < selectedGenres.Count(); i++)
            {
                // cartii pe care vrem sa o adaugam in baza de date ii
                // asignam genurile selectate
                Genre genre = db.Genres.Find(selectedGenres[i].Id);
                bookRequest.Genres.Add(genre);
            }
            db.Books.Add(bookRequest);
            db.SaveChanges();
            return RedirectToAction("Index");
        }
        return View(bookRequest);
    }
    catch(Exception e)
    {
        var msg = e.Message;
        return View(bookRequest);
    }
}
```

Editarea unei carti

Functionalitate: Atunci cand editam o carte existenta in BD vrem sa ne apara in interiorul formularului o lista de checkbox-uri cu toate genurile din BD. Iar cele care apartin cartii sa fie bifate atunci cand accesam pagina formularului.

Pasul 1: Vom modifica actiunile **Edit** de tipul GET si PUT din **BookController.cs.**

```
        [HttpGet]
        public ActionResult Edit(int? id)
        {
            if (id.HasValue)
            {
                Book book = db.Books.Find(id);
                book.BookTypeList = GetAllBookTypes();
                book.PublisherList = GetAllPublishers();
                book.GenresList = GetAllGenres();

                foreach(Genre checkedGenre in book.Genres)
                {   // iteram prin genurile care erau atribuite cartii inainte de momentul accesarii
formularului
                    // si le selectam/bifam  in lista de checkbox-uri
                    book.GenresList.FirstOrDefault(g => g.Id == checkedGenre.GenreId).Checked = true;
                }
                if (book == null)
                {
                    return HttpNotFound("Coludn't find the book with id " + id.ToString() + "!");
                }
                return View(book);
            }
            return HttpNotFound("Missing book id parameter!");
        }

        [HttpPut]
        public ActionResult Edit(int id, Book bookRequest)
        {
            bookRequest.BookTypeList = GetAllBookTypes();
            bookRequest.PublisherList = GetAllPublishers();

            // preluam cartea pe care vrem sa o modificam din baza de date
            Book book = db.Books.Include("Publisher").Include("BookType")
                        .SingleOrDefault(b => b.BookId.Equals(id));

            // memoram intr-o lista doar genurile care au fost selectate din formular
            var selectedGenres = bookRequest.GenresList.Where(b => b.Checked).ToList();
            try
            {
                if (ModelState.IsValid)
                {
                    if (TryUpdateModel(book))
                    {
                        book.Title = bookRequest.Title;
                        book.Author = bookRequest.Author;
                        book.Summary = bookRequest.Summary;
                        book.NoOfPages = bookRequest.NoOfPages;
                        book.PublisherId = bookRequest.PublisherId;
                        book.BookTypeId = bookRequest.BookTypeId;

                        book.Genres.Clear();
                        book.Genres = new List<Genre>();

                        for (int i = 0; i < selectedGenres.Count(); i++)
                        {
                            // cartii pe care vrem sa o editam ii asignam genurile selectate
                            Genre genre = db.Genres.Find(selectedGenres[i].Id);
                            book.Genres.Add(genre);
                        }
```

```
                    db.SaveChanges();
                }
                return RedirectToAction("Index");
            }
            return View(bookRequest);
        }
        catch (Exception)
        {
            return View(bookRequest);
        }
    }
```

Pasul 2: In **Views/New.cshtml** si **Views/Edit.cshtml:**

```
@Html.LabelFor(b => b.GenresList, "Choose genres:")
<br />
for(int i = 0; i < Model.GenresList.Count(); i++)
{
    @Html.HiddenFor(b => b.GenresList[i].Id)
    @Html.HiddenFor(b => b.GenresList[i].Name)
    @Html.CheckBoxFor(b => b.GenresList[i].Checked)
    @Html.DisplayFor(b => b.GenresList[i].Name)
    <br />
}

<br />
```

!!!!! ATENTIE !!!! Nu uitati sa puneti **@Html.HiddenFor si pentru campul nume**. Daca nu ati pus aceasta linie de cod si suntei in cazul in care modelul nu poate fi adaugat/editat in/din BD din cauza erorilor generate de validari sau de model bindig, iar utilizatorul ramane blocat pe pagina formularului (adica, se executa un refresh), veti vedea ca pe ecran vor aparea doar casutele checkbox-urilor, fara numele genurilor.

Daca rulati in modul debug veti putea vedea ca orice element din lista de checkbox-uri va avea doar campul `Name = null`.

Acest lucru se intampla din cauza ca **@Html.Display** este corespondentul tag-ului **<p>** din HTML, care doar afiseaza datele pe ecran.

Iar, **`@Html.HiddenFor(b => b.GenresList[i].Name)`**, unde **`i = 0`** este corespondentul tag-ului

```
<input data-val="true"
            id="GenresList_0__Name"
            name="GenresList[0].Name"
            type="hidden"
            value="Horror" />   // value = valoarea lui b.GenresList[0].Name
```

din HTML, care introduce numele checkbox-ului intr-un field ascuns.