Dezvoltarea Aplicatiilor Web utilizand ASP.NET Core MVC
Laborator 7

EXERCITII:

Se considera baza de date, cu cele trei modele Article.cs, Category.cs si Comment.cs din Laborator 6. Modificati atat View-urile, cat si metodele din Controllere (atunci cand este necesar) astfel incat sa se utilizeze helpere pentru View. Pentru exemple – **VEZI Curs 7 Sectiunea Helpere pentru View.**

1. Studiati sectiunea **Trimiterea datelor catre View → Model** din cadrul Cursului 7, dupa care implementati urmatoarea functionalitate:

    a. Sa se modifice actiunea (metoda) **Show** din Controller-ul Articles, astfel incat sa se trimita Modelul de tip Article catre View-ul asociat. Pasul urmator este modificarea View-ului, astfel incat sa includa modelul si sa afiseze proprietatile acestuia catre utilizatorul final.

2. Modificati functionalitatea de **adaugare a unui nou articol** in baza de date, impreuna cu selectarea categoriei din care face parte, astfel:

    a. **Metoda New cu Get** o sa apeleze o metoda, cu ajutorul careia se vor prelua categoriile din baza de date, in vederea trimiterii acestora in View-ul asociat pentru a popula un element de tip dropdown. Este necesar un dropdown deoarece in momentul in care utilizatorul final doreste sa adauge un nou articol in baza de date, acesta trebuie sa aleaga si o categorie din care o sa faca parte articolul respectiv. Alegerea categoriei o sa se realizeze cu ajutorul un dropdown. **Ce fel de metoda trebuie sa fie metoda prin care se preiau toate categoriile din baza de date?**

```
        public IEnumerable<SelectListItem> GetAllCategories()
        {
            // generam o lista de tipul SelectListItem fara elemente
            var selectList = new List<SelectListItem>();

            // extragem toate categoriile din baza de date
            var categories = from cat in db.Categories
                             select cat;

            // iteram prin categorii
            foreach (var category in categories)
            {
                // adaugam in lista elementele necesare pentru
dropdown
                // id-ul categoriei si denumirea acesteia
                selectList.Add(new SelectListItem
                {
                    Value = category.Id.ToString(),
                    Text = category.CategoryName.ToString()
                });
            }

                /* Sau se poate implementa astfel:
                 *
                foreach (var category in categories)
                {
                    var listItem = new SelectListItem();
                    listItem.Value = category.Id.ToString();
                    listItem.Text = category.CategoryName.ToString();

                    selectList.Add(listItem);
                 }*/


            // returnam lista de categorii
            return selectList;
        }
```

Implementare dropdown:

Dupa implementarea metodei anterioare, prin care sunt preluate toate categoriile din baza de date, urmeaza apelarea acesteia in metoda New din Controller-ul Articles si stocarea categoriilor cu ajutorul unei proprietati, dupa cum urmeaza:

```
public IActionResult New()
   {

       Article article = new Article();

       article.Categ = GetAllCategories();

       return View(article);
   }
```

- Se instantiaza clasa Article deoarece in cadrul formularului o sa se creeze un nou obiect in baza de date de tipul modelului Article;

- Se apeleaza metoda `GetAllCategories()` prin care sunt preluate toate categoriile din baza de date si se stocheaza aceste categorii pentru a fi trimise catre View-ul New;

- Stocarea listei care contine toate categoriile din baza de date (de forma id_categorie – denumire categorie) se realizeaza prin adaugarea unei noi proprietati in Modelul Article:

```
[NotMapped]
public IEnumerable<SelectListItem> Categ { get; set; }
```

Cu ajutorul acestui atribut se pot prelua categoriile in cadrul Helper-ului `@Html.DropDownListFor` astfel:

```
@Html.DropDownListFor(m => m.CategoryId, new
SelectList(Model.Categ, "Value", "Text"), "Selectati
categoria", new { @class = "form-control" })
```

- Se trimite un obiect de tipul modelului Article catre View-ul asociat, astfel incat View-ul sa primeasca acest model prin intermediul Helperului @model

Folosind Helper-ul **@model** in View pentru modelul Article, putem adauga **@Html.DropDownListFor** pentru acest model. Acest Helper (DropDownListFor) primeste urmatorii parametri:

- Primul parametru este o lambda expresie prin care se alege atributul modelului pentru care se va genera elementul de tip select (in acest caz pentru id-ul categoriei → **CategoryId**);
- Al doilea parametru trebuie sa fie o lista de tipul **SelectList** cu elementele pentru care se va genera acest dropdown
- Al treilea element reprezinta o optiune default (Selectati categoria)
- Al patrulea element este optional si reprezinta o lista de atribute aditionale pentru acest element generat

View-ul New – implementare utilizand Helpere (`@Html.Label, @Html.TextBox, @Html.TextArea, @Html.DropDownListFor`):

```
@model ArticlesApp.Models.Article

<h2 class="text-center mt-5">Adaugare articol</h2>
<br />
<div class="container mt-5">
    <div class="row">
        <div class="col-6 offset-3">

            <form method="post" action="/Articles/New">

                @Html.Label("Title", "Titlu Articol")
                <br />

                @Html.TextBox("Title", null, new { @class = "form-
control" })
```

```
                <br /><br />

                @Html.Label("Content", "Continut Articol")
                <br />
                @Html.TextArea("Content", null, new { @class = "form-
control" })
                <br /><br />

                <label>Selectati categoria</label>
                @Html.DropDownListFor(m => m.CategoryId, new
SelectList(Model.Categ, "Value", "Text"),
                "Selectati categoria", new { @class = "form-control"
})
                <br />

                <button class="btn btn-success" type="submit">Adauga
articol</button>

            </form>
        </div>
    </div>
</div>
```

b. **Metoda New cu POST** – sa se adauge articolul in baza de date si sa se afiseze un mesaj sugestiv “Articolul a fost adaugat”. Pentru afisarea mesajului o sa se utilizeze o variabila de tip TempData conform **Curs 7 – Sectiunea Trimiterea Datelor catre View -> TempData**.

   Ce trebuie sa returnam in cazul in care adaugarea articolului in baza de date nu s-a putut realiza cu succes (pe ramura catch())?

3. Modificati functionalitatea de **editare a unui articol existent** in baza de date, impreuna cu selectarea categoriei din care face parte, astfel:
   a. Actiunea Edit cu GET sa foloseasca metoda `GetAllCategories()` pentru preluarea categoriilor din baza de date si popularea unui element de tip `@Html.DropDownListFor`
   b. Actiunea Edit cu GET o sa trimita catre View-ul asociat un model de tip Article
   c. View-ul Edit trebuie sa contina Helpere pentru View (`@Html.Label, @Html.EditorFor, @Html.DropDownListFor`)

```
@model ArticlesApp.Models.Article

<form method="post" action="/Articles/Edit/@Model.Id">


                @Html.Label("Title", "Titlu Articol")
                <br />
                @Html.EditorFor(m => m.Title, new { htmlAttributes =
new { @class = "form-control" } })
                <br /><br />

                @Html.Label("Content", "Continut Articol")
                <br />
                @Html.Editor("Content", new { htmlAttributes = new {
@class = "form-control" } })
                <br /><br />

                <label>Selectati categoria</label>
                @Html.DropDownListFor(m => m.CategoryId,
                new SelectList(Model.Categ, "Value", "Text"),
                "Selectati categoria", new { @class = "form-control"
})
                <br />

                <button class="btn btn-sm btn-success"
type="submit">Modifica articol</button>

</form>
```

d. **Actiunea Edit cu POST** sa realizeze adaugarea modificarilor in baza de date si sa se afiseze un mesaj corespunzator: “Articolul a fost modificat”, utilizand o variabila de tip TempData.

4. Modificati functionalitatea de stergere a unui articol din baza de date, astfel incat in metoda Delete, dupa realizarea operatiei de stergere, sa se afiseze mesajul: “Articolul a fost sters”, utilizand o variabila de tip TempData.

5. Sa se procedeze la fel ca in implementarile anterioare si pentru entitatea Category (utilizarea helperelor pentru View si a mesajelor de tip TempData pentru afisarea mesajelor sugestive catre utilizatorul final).