Dezvoltarea Aplicatiilor Web utilizand ASP.NET Core MVC
Laborator 8

EXERCITII:

Se considera baza de date, cu cele trei modele Article.cs si Category.cs, Comment.cs, din laboratorul anterior.

Sa se modifice implementarea, acolo unde este necesar, astfel incat sa fie posibila integrarea validarilor necesare. **Cititi cu atentie notiunile din cadrul Cursului 8 si implementati exercitiile urmatoare.**

Sugestii de implementare:

1. Sa se modifice Modelele (Article, Category, Comment) adaugandu-se validarile necesare la nivel de Model astfel (**VEZI Curs 8 – Sectiunea Atribute de validare la nivel de Model**):

- Se adauga asupra modelului **Article** urmatoarele validari:
  - **Titlul** articolului este obligatoriu (Required), poate avea o lungime maxima de 100 de caractere (StringLength) si nu poate avea mai putin de 5 caractere (MinLength)
  - **Continutul** articolului este obligatoriu (Required)
  - **Categoria** din care face parte articolul este obligatorie (Required)

- Se adauga asupra modelului **Category** urmatoarea validare:
  - **Numele** categoriei este obligatoriu (Required)

- Se adauga asupra modelului **Comment** urmatoarea validare:
  - **Continutul** comentariului este obligatoriu (Required)

2. Preluati validarile in View-urile asociate (**VEZI Curs 8 – Sectiunea Preluarea validarilor in View**) urmand pasii urmatori. De asemenea, dupa adaugarea validarilor in View, studiati cu atentie **Exemplul 2 din Cursul 8**. In cadrul exemplului se observa de ce mesajele de eroare nu se afiseaza corect si cum se poate rezolva aceasta problema. Implementati corect mesajele de eroare in cadrul aplicatiei *ArticlesApp*.

   - Se preiau validarile pentru entitatea Article, View-urile New si Edit (adaugarea unui nou articol si editarea unui articol existent)

   - Validarile se includ in View cu ajutorul Helper-ului specific `@Html.ValidationMessageFor` sau `@Html.ValidationMessage` cele doua fiind echivalente (**VEZI Curs 8 – Sectiunea @Html.ValidationMessage si @Html.ValidationMessageFor**)

   - Inlocuiti tagul **`<form>`** utilizat pentru crearea unui formular, cu Helper-ul specific (**VEZI Curs 8 – Sectiunea Helper-ul @Html.BeginForm**)

   - Afisati un sumar cu toate erorile aparute in timpul validarii (**VEZI Curs 8 – Sectiunea Helper-ul @Html.ValidationSummary**)

3. Implementati validarile la nivel de Controller, astfel incat acestea sa se afiseze corect in functie de actiunea pe care o face utilizatorul final (**VEZI Curs 8 – Sectiunea Implementarea validarilor la nivel de Controller**)

Explicatii privind adaugarea validarilor la nivel de Controller:

In metoda **New** cu **HttpPost** trebuie sa adaugam si verificarea starii modelului. Prin intermediul variabilei **ModelState** se verifica daca toate validarile au trecut cu succes si se pot salva modificarile in baza de date.

```
[HttpPost]
public IActionResult New(Article article)
{
      article.Date = DateTime.Now;
      article.Categ = GetAllCategories();

       if (ModelState.IsValid)
       {
            db.Articles.Add(article);
            db.SaveChanges();
            TempData["message"] = "Articolul a fost
adaugat";
            return RedirectToAction("Index");
      }
      else
      {
            return View(article);
      }
}
```

In momentul in care validarile nu trec cu succes, ramura de executie va fi cea din **else**. Astfel, in momentul in care returnam View-ul este necesar sa trimitem din nou modelul impreuna cu toate atributele sale.

- **La fel se procedeaza si pentru editare**

```
[HttpPost]
public IActionResult Edit(int id, Article requestArticle)
{
      Article article = db.Articles.Find(id);
      requestArticle.Categ = GetAllCategories();

      if(ModelState.IsValid)
      {
            article.Title = requestArticle.Title;
            article.Content = requestArticle.Content;
            article.CategoryId = requestArticle.CategoryId;
            TempData["message"] = "Articolul a fost modificat";
            db.SaveChanges();
            return RedirectToAction("Index");
      }
      else
      {
            return View(requestArticle);
      }
}
```

4. Pentru o mentenanta mai buna a codului si pentru a elimina redundanta, utilizati un Partial View pentru afisarea informatiilor asociate unui articol.

   In cadrul aplicatiei, afisarea unui articol contine acelasi cod, atat in cazul afisarii tuturor articolelor din baza de date → View-ul Index, cat si in cazul afisarii unui singur articol → View-ul Show. In acest caz, pentru a elimina redundanta, si anume scrierea repetitiva a aceluiasi cod in cadrul ambelor View-uri, sa se utilizeze un View Partial numit ***ArticleInfo*** (**VEZI Curs 8 – Sectiunea View-uri partajate – Partial View**).

5. Sa se modifice Layout-ul existent, astfel incat sa contina link-uri atat catre pagina de afisarea a tuturor articolelor, catre pagina de afisare a tuturor categoriilor si catre pagina de adaugare a unui nou articol. O sa se modifice Layout-ul existent, numit _Layout.cshtml, aflat in folderul View → folderul Shared. In cadrul acestui Layout o sa se adauge cele trei link-uri. De asemenea, o sa se modifice link-ul existent in Layout, cel care la apasarea logo-ului duce la /Home/Index. Dupa modificare, acesta trebuie sa redirectioneze catre /Articles/Index.

Pentru stilizare se pot modifica clasele existente de css, adaugand urmatoarele secvente de cod in **wwwroot → css → site.css**

Sugestii de stilizare (pentru paleta de culori se poate utiliza: https://flatuicolors.com/)

```
.navbar {
    background-color: #dfe6e9 !important;
    border: none !important;
    box-shadow: 0 3px 12px #636e72 !important;
}

a.nav-link {
    color: #2c2c2c !important;
}
```

```
a.nav-link:hover {
    color: #00b894 !important;
}

.logoutbtn {
    color: #2c2c2c !important;
}

.logoutbtn:hover {
    color: #00b894 !important;
}
```

6. Sa se adauge validarile si pentru entitatile **Category** si **Comment**.

- **In Model** – Denumirea categoriei este obligatorie si Continutul comentariului este obligatoriu
- **In View** – sa se utilizeze Helper-ul specific pentru validare
- **In Controller** – sa se verifice starea modelului (se verifica daca validarile au trecut cu succes)

/!\ OBSERVATIE

In cazul adaugarii unui comentariu, formularul de adaugare se afla in View-ul Show asociat Controller-ului Articles. Am procedat in acest mod deoarece un comentariu este asociat unui articol si este necesara afisarea acestuia impreuna cu articolul de care depinde.

Asadar, in pagina Show a articolului, se afiseaza articolul impreuna cu toate comentariile sale, dar si cu un formular in care utilizatorul poate adauga un nou comentariu articolului respectiv. Implementarea formularului este urmatoarea:

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

     <div class="card-body">

          <input type="hidden" name="ArticleId"
value="@Model.Id" />

          <label>Continut comentariu</label>
          <br />

          <textarea class="form-control"
name="Content"></textarea>

           @Html.ValidationMessage("Content", null, new {
@class = "text-danger"})

          <br /><br />

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

     </div>

</form>
```

In cadrul formularului, am adaugat in acest caz si validarea cu ajutorul Helperului specific.

De asemenea, se poate observa si inputul de tip ***hidden*** cu ajutorul caruia retinem id-ul articolului caruia ii corespunde comentariul, conform implementarii modelului (se observa cheia externa ArticleId):

```
public class Comment
    {
        [Key]
        public int Id { get; set; }

        [Required(ErrorMessage = "Continutul comentariului este
obligatoriu")]
        public string Content { get; set; }

        public DateTime Date { get; set; }

        public int? ArticleId { get; set; }

        public virtual Article? Article { get; set; }
    }
```

Avand in vedere ca un View nu poate avea mai mult de un Model inclus, nu se pot utiliza Helpere pentru restul componentelor de HTML (input, label, textarea). In acest caz, se pot utiliza in continuare tag-urile de HTML.

Deoarece dorim ca in momentul in care se incearca adaugarea unui comentariu fara continut, utilizatorul sa primeasca mesajul de validare "Comentariul este obligatoriu", dar in acelasi timp trebuie ca pe ramura *else* sa retrimitem intregul articol impreuna cu toate comentariile pe care le are, trebuie sa procedam astfel:

- O sa mutam metoda New cu HttpPost din CommentsController in ArticlesController

- In ArticlesController metoda o sa se numeasca Show deoarece dorim ca in momentul in care nu se poate adauga comentariul, sa redirectioneze in metoda Show a articolului, astfel incat sa se afiseze corect articolul impreuna cu toate comentariile sale. Se poate proceda in acest mod atunci cand dorim sa pastram obiectul si implicit trebuie sa-l pasam ca argument in View. In cazul nostru, obiectul este de tip Article si trebuie pasat ca argument in View-ul Show pentru afisarea articolului impreuna cu toate comentariile, dar si cu acel formular in care se poate adauga un nou comentariu. Avand in vedere ca in ArticlesController mai avem o metoda Show, metoda pe care dorim sa o implementam pentru adaugarea unui nou comentariu se poate numi Show, dar are nevoie de un verb Http diferit. Adaugarea unui comentariu fiind o actiune de scriere, o sa aiba verbul HttpPost.

- Avand in vedere ca in Show modelul de baza este Article, atunci cand se incearca adaugarea comentariului, se preia si id-ul articolului (id-ul modelului principal din view). Pentru a prelua doar datele comentariului si pentru inserarea corecta a acestora in baza de date, se utilizeaza in metoda Show [FromForm] astfel incat datele sa fie preluate doar din formular

```
public IActionResult Show([FromForm] Comment comment)
```

```
        [HttpPost]
        public IActionResult Show([FromForm] Comment comment)
        {
            comment.Date = DateTime.Now;

            if (ModelState.IsValid)
            {
                db.Comments.Add(comment);
                db.SaveChanges();
                return Redirect("/Articles/Show/" +
comment.ArticleId);
            }

            else
            {
               Article art =
     db.Articles.Include("Category").Include("Comments")
                                .Where(art => art.Id ==
comment.ArticleId)
                                .First();

                //return Redirect("/Articles/Show/" +
comm.ArticleId);

                return View(art);
            }
        }
```

Daca sursa default care trimite datele nu paseaza corect argumentele la nivel de Controller, se pot utiliza urmatoarele atribute pentru a specifica sursa preluarii valorilor:

- **[FromQuery]** – preia valorile din query-ul de baze de date
- **[FromRoute]** – preia valorile din ruta
- **[FromForm]** – preia valorile din formular
- **[FromBody]** – prea valorile din corpul request-ului (atunci cand se utilizeaza JSON)
- **[FromHeader]** – preia valorile din headerele Http (fiecare request are headere Http care pot fi interpretate de catre server/aplicatie)