Dezvoltarea Aplicatiilor Web-Anul 3
Laborator 10

Exercitiul 1

Se consideram urmatorul exemplu:

O aplicatie in care se pot adauga articole de catre utilizatori (editori). Fiecare articol are o singura categorie. Categoriile pot fi administrate doar de utilizatorii cu rolul "Administrator". Editorii pot sa adauge, listeze, modifice si stearga **doar** articolele *proprii*. Utilizatorii inregistrati (fara rol de "Editor" – sau cu rolul "User") pot doar sa vizualizeze articolele existente. Utilizatorii neinregistrati nu pot vedea articolele. Acestia sunt redirectionati catre pagina de inregistrare sau autentificare.

Sa se implementeze sistemul de autentificare
(Vezi Curs 10, paginile 1-5).

Sa se copieze fisierele corespunzatoare din proiectul realizat in laboratorul 9 in acest proiect. In folderul Controllers se copiaza **ArticleController** si **CategoriesController.** In folderul Models – modelele **Article.cs** si **Category.cs.** In Views se copiaza folderele **Article** si **Categories.** In folderul Shared se copiza partialele **ArticleInfo.cshtml** si **ShowPartial.cshtml.** In Global.asax o sa avem nevoie de urmatoarea secventa de cod pentru momentul in care o sa modificam intr-un anumit model.

```
    Database.SetInitializer<ApplicationDbContext>(new
DropCreateDatabaseIfModelChanges<ApplicationDbContext>());
```

Pentru adaugarea de roluri si pentru realizarea asocierii dintre utilizatori si roluri trebuie parcursi pasii din Cursul 10, pagina 6, adaugand secventele de cod din curs in folderul **App_Start**, in fisierul **Startup.Auth.cs** si in folderul **App_Start**, in fisierul **IdentityConfig.cs.**

In fisierul **Startup.cs** se defineste o metoda care se apeleaza in corpul functiei **Configuration**. Aceasta metoda va contine codul necesar pentru adaugarea rolurilor si a utilizatorilor (Vezi Curs 10, paginile 6-7).

De asemenea, contextul bazei de date, se va muta in modelul **IdentityModel.cs** deoarece este necesar ca tabelele utilizatorilor, cat si tabelele celorlalte modele din aplicatie sa se regaseasca in aceeasi baza de date.

```
public class ApplicationDbContext : IdentityDbContext<ApplicationUser>
{
    public ApplicationDbContext()
        : base("DefaultConnection", throwIfV1Schema: false)
    {
    }
    public DbSet<Article> Articles { get; set; }
    public DbSet<Category> Categories { get; set; }
    public static ApplicationDbContext Create()
    {
        return new ApplicationDbContext();
    }
}
```

Pentru a restrictiona accesul utilizatorilor la metodele pentru C.R.U.D. pe categorii, vom folosi helper-ul Authorize la nivel de Controller pentru controller-ul Categories:

```
[Authorize(Roles = "Administrator")]
    public class CategoriesController : Controller
    {
        private ApplicationDbContext db = ApplicationDbContext.Create();
      ...
```

Pentru rolul de editor trebuie facute mai multe modificari. In primul rand, se adauga pentru metodele **New, Edit** si **Delete** urmatoarele:

- Pentru metodele **Index, Show** vom folosi:
  ```
  [Authorize(Roles = "User,Editor,Administrator")]
  ```

- Pentru metodele **New, Edit, Delete** vom folosi:
  ```
  [Authorize(Roles = "Editor,Administrator")]
  ```

Dupa alocarea rolurilor este necesar sa modificam modelul **Article** (modelul corespunzator pentru articole) astfel incat sa includa si ID-ul utilizatorului care a creat respectivul articol:

```
. . .
    public string UserId { get; set; }
    public virtual ApplicationUser User { get; set; }
```

Metodele Edit si New se modifica corespunzator astfel incat la inserarea in baza de date si la modicarea unui articol sa se tina cont de ID-ul utilizatorului care face aceasta actiune.

La inserarea in baza de date trebuie transmis ca parametru, prin intermediul formularului, si ID-ul utilizatorui. Astfel, el va fi pus in variabila ViewBag din Controller si va fi inclus sub forma unui camp ascuns in View (Vezi Curs 10, paginile 10-14).

/! La final modificam codul existent in Show, atat in Controller, cat si in View, astfel incat butoanele de Edit si Delete sa fie vizibile doar pentru Administrator si Editor. Administratorul poate avea butoanele vizibile pentru orice articol, iar Editorul o sa le aiba vizibile doar pentru articolele care ii apartin. Utilizatorul cu rolul User nu o sa vada aceste butoane.

ArticleController – Show

```
[Authorize(Roles = "User,Editor,Administrator")]
        public ActionResult Show(int id)
        {
            Article article = db.Articles.Find(id);
            ViewBag.Article = article;
            ViewBag.Category = article.Category;
            ViewBag.afisareButoane = false;
            if(User.IsInRole("Editor") || User.IsInRole("Administrator")){
                ViewBag.afisareButoane = true;
            }
            ViewBag.esteAdmin = User.IsInRole("Administrator");
            ViewBag.utilizatorCurent = User.Identity.GetUserId();
            return View(article);
         }
```

Show.cshtml

```
@model CursLab8.Models.Article

@{
    ViewBag.Titlu = "Afisare articol";
}

<h2>@ViewBag.Titlu</h2>

<br />

@Html.Partial("ShowPartial")

@if (ViewBag.afisareButoane == true && Model.UserId ==
ViewBag.utilizatorCurent || ViewBag.esteAdmin)
{
    <a class="btn btn-sm btn-info" href="/Article/Edit/@Model.Id">
        Modifica articol
    </a>
    <br />

    <form method="post" action="/Article/Delete/@Model.Id">
        @Html.HttpMethodOverride(HttpVerbs.Delete)
        <button class="btn btn-sm btn-danger" type="submit">Sterge
articol</button>
    </form>
}

<hr />
<a class="btn btn-sm btn-success" href="/Article/Index">Inapoi la articole</a>
<a class="btn btn-sm btn-success" href="/Article/New">Adauga articol</a>

<br />
```