Dezvoltarea Aplicatiilor Web-Anul 2
Laborator 8

Exercitiul 1

Se considera urmatorul exemplu:

O aplicatie in care se pot adauga articole de catre utilizatori – cu rolul "**Editor**". 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. De asemenea, toti utilizatorii pot sa vada comentariile asociate unui articol, sa lase comentarii, dar si sa editeze sau sa stearga propriile comentarii.

Sa se implementeze **sistemul de autentificare** (Vezi **Curs 8,** sectiunea **Autentificare**).

Pentru **adaugarea rolurilor** si pentru realizarea asocierii dintre utilizatori si roluri (**Vezi Curs 8**, sectiunea Roluri. Asocierea dintre roluri si utilizatori). Astfel, se adauga secventele de cod din curs in folderul **App_Start**, in fisierul **Startup.Auth.cs** si in folderul **App_Start -> 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.

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)
    {
         Database.SetInitializer(new
        MigrateDatabaseToLatestVersion<ApplicationDbContext,
        Laborator8App.Migrations.Configuration>("DefaultConnection"));
    }

    public DbSet<Article> Articles { get; set; }
    public DbSet<Category> Categories { get; set; }
    public DbSet<Comment> Comments { get; set; }

    public static ApplicationDbContext Create()
    {
        return new ApplicationDbContext();
    }
}
```

Sa se copieze fisierele corespunzatoare din proiectul realizat in **laboratorul 7**, in acest proiect. In folderul **Controllers** se copiaza **ArticlesController** si **CategoriesController si CommentsController.** In folderul **Models** – modelele **Article.cs** si **Category.cs si Comment.cs.** In **Views** se copiaza folderele **Article**, **Category si Comment.** In folderul Shared se copiaza partialul **ArticleInfo.cshtml.**

Dupa ce se adauga clasele se va activa sistemul de migratii.

/!\ OBSERVATIE

Pentru a aloca in mod automat unui utilizator un rol la ***inregistrare*** este necesar sa se modifice fisierul **AccountController** din folderul Controllers. In metoda **Register ([HttpPost])** trebuie adaugata secventa de cod necesara pentru asocierea utilizatorului cu rolul dorit in momentul salvarii:

```
        // POST: /Account/Register
          [HttpPost]
          [AllowAnonymous]
          [ValidateAntiForgeryToken]
          public async Task<ActionResult> Register(RegisterViewModel model)
          {
              if (ModelState.IsValid)
              {
                  var user = new ApplicationUser { UserName = model.Email,
Email = model.Email };
                  var result = await UserManager.CreateAsync(user,
model.Password);
                  if (result.Succeeded)
                  {
                      await SignInManager.SignInAsync(user, isPersistent:false,
rememberBrowser:false);

                      UserManager.AddToRole(user.Id, "User");
                    . . .
          }
```

In folderul Shared -> _Layout.cshtml (pagina de Layout a aplicatiei) se modifica astfel incat in meniul din partea de sus a paginii sa existe: numele aplicatiei care sa redirectioneze paginile de fiecare data catre pagina de afisare a tuturor articolelor, cat si alte trei link-uri: “Afisare articole”, “Afisare categorii” si “Adaugare Articol”.

Pentru stilizarea meniului se pot adauga clase de CSS in Content -> Site.css.

De exemplu, se poate stiliza elementul “navbar” astfel:

```
.navbar {
    background-color: #dadada;
    border: none;
    box-shadow: 0 6px 12px #dadada;
}

.navbar a {
    color: #2c2c2c;
}

.navbar a:hover {
    color: #5cb85c;
}
```

Dupa configurarea sistemului de autentificare si a tuturor rolurilor din aplicatie ne putem inregistra astfel incat sa avem 3 conturi – cate un cont pentru fiecare rol. Pentru inceput se pot asocia userii cu rolurile direct in baza de date. De asemenea, se vor adauga si cateva intrari in baza de date.

Pentru a **restrictiona accesul utilizatorilor** la metodele pentru C.R.U.D. pe categorii (doar userul cu rolul “Administrator” poate face operatii C.R.U.D pe categorii), vom folosi helper-ul **Authorize** la nivel de Controller pentru Controller-ul **Categories**:

```
[Authorize(Roles = "Admin")]
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 existente urmatoarele:

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

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

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 utilizatorului. Astfel, el va fi pus in variabila ViewBag din Controller si va fi inclus sub forma unui camp ascuns in View.

```
...
    [Authorize(Roles = "Editor,Admin")]
    public ActionResult New()
    {
        Article article = new Article();
        // preluam lista de categorii din metoda GetAllCategories()
        article.Categories = GetAllCategories();

        // Preluam ID-ul utilizatorului curent
        article.UserId = User.Identity.GetUserId();
        return View(article);
    }
...
```

La fel se pastreaza si in cazul metodei New cu HttpPost

```
[HttpPost]
[Authorize(Roles = "Editor,Admin")]
public ActionResult New(Article article)
{
    article.Date = DateTime.Now;
    article.UserId = User.Identity.GetUserId();
    …
}
```

Pentru afisarea tuturor informatiilor referitoare la un articol (de exemplu in metoda Index) este necesar sa facem Join si cu modelul User (ApplicationUser din fisierul IdentityModels.cs) astfel:

```
var articles = db.Articles.Include("Category").Include("User");
```

View-ul se va modifica pentru a include si numele autorului articolului. Se poate adauga secventa de cod chiar in partialul “ArticleInfo”:

```
...
<i class="glyphicon glyphicon-user"></i> <i>articol scris de</i>
<strong> @Model.User.UserName </strong>
...
```

Urmatorul pas necesar este de a verifica daca ID-ul utilizatorului care doreste sa editeze, respectiv sa stearga un articol, corespunde cu ID-ul utilizatorului care a creat articolul respectiv. De asemenea, daca utilizatorul care incearca aceste actiuni are rolul “Admin” atunci verificarea nu mai este necesara

```
[Authorize(Roles = "Editor,Admin")]
public ActionResult Edit(int id)
{
    Article article = db.Articles.Find(id);
    article.Categories = GetAllCategories();

   if(article.UserId == User.Identity.GetUserId() ||
   User.IsInRole("Admin"))
    {
        return View(article);
    } else
    {
        TempData["message"] = "Nu aveti dreptul sa faceti modificari
asupra unui articol care nu va apartine!";
        return RedirectToAction("Index");
    }
}

. . .
[HttpPut]
[Authorize(Roles = "Editor,Admin")]
public ActionResult Edit(int id, Article requestArticle)
{
    try
    {
        if (ModelState.IsValid)
        {
            Article article = db.Articles.Find(id);
```

```
                if (article.UserId == User.Identity.GetUserId() ||
                User.IsInRole("Administrator"))
                {
                    if (TryUpdateModel(article))
                    {
                        article.Title = requestArticle.Title;
                        article.Content = requestArticle.Content;
                        article.Date = requestArticle.Date;
                        article.CategoryId = requestArticle.CategoryId;
                        db.SaveChanges();
                        TempData["message"] = "Articolul a fost
modificat!";
                    }
                    return RedirectToAction("Index");
                }
                else
                {
                    TempData["message"] = "Nu aveti dreptul sa faceti
modificari asupra unui articol care nu va apartine!";
                    return RedirectToAction("Index");
                }
            }
            else
            {
                return View(requestArticle);
            }
        }
        catch (Exception e)
        {
            return View(requestArticle);
```

```
        }
    }
    . . .

    [HttpDelete]
    [Authorize(Roles = "Editor,Admin")]
    public ActionResult Delete(int id)
    {
        Article article = db.Articles.Find(id);
        if (article.UserId == User.Identity.GetUserId() ||
        User.IsInRole("Administrator"))
        {
            db.Articles.Remove(article);
            db.SaveChanges();
            TempData["message"] = "Articolul a fost sters!";
            return RedirectToAction("Index");
        }
        else
        {
            TempData["message"] = "Nu aveti dreptul sa stergeti un
articol care nu va apartine!";
            return RedirectToAction("Index");
        }
    }
```

/!\ 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 Admin 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.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 – exemplu:

```
<div class="panel panel-default">
    @Html.Partial("ArticleInfo", Model)
    @if (ViewBag.afisareButoane == true && Model.UserId == ViewBag.utilizatorCurent ||
ViewBag.esteAdmin)
    {
        <div class="panel-footer">
            <a class="btn btn-success pull-left" href="/Articles/Edit/@Model.Id">Modifica
articol</a>

            @using (Html.BeginForm(actionName: "Delete", controllerName: "Articles",
method: FormMethod.Post, routeValues: new { id = @Model.Id }))
            {
                @Html.HttpMethodOverride(HttpVerbs.Delete)
                <button class="btn btn-danger pull-right" type="submit">Sterge
articol</button>
            }
            <div class="clearfix"></div>
        </div>
    }

</div>
```

In final, modificam paginile Index si Show, astfel incat sa nu mai existe legatura catre paginile de afisare a tuturor articolelor, a tuturor categoriilor sau catre pagina de adaugare a unui nou articol. Vom proceda astfel: aceste legaturi o sa fie adaugate in meniul din Shared -> _Layout.cshtml.

Utilizatorii normali cu rolul “User” o sa vada in meniu doar link-ul “Afisare articole”, utilizatorii cu rolul “Editor” o sa vada in meniu link-urile “Afisare articole” si “Adaugare articol”, iar userul cu rolul “Admin” o sa vada in aplicatie “Afisare articole”, “Adaugare articol”, “Afisare categorii”.

Se utilizeaza in pagina _Layout.cshtml urmatoarele verificari, dupa caz:

```
@if (User.IsInRole("Admin")) …

@if (User.IsInRole("Editor") …
```