

Dezvoltarea Aplicatiilor Web-Anul 2

Laborator 10

Exercitiu:

Stilizati aplicatia dezvoltata in laboratoarele precedente astfel:

- ✚ **Meniul** (Views -> Shared -> _Layout.cshtml) va afisa link-urile in functie de rolul pe care il are utilizatorul:
 - **Utilizatorii neautentificati** nu o sa vada decat logo-ul aplicatiei (la apasarea acestuia, utilizatorii o sa fie redirectionat catre pagina Index din HomeController)
 - **Utilizatorul cu rolul Admin** poate accesa “Afisare articole”, “Afisare categorii”, “Afisare utilizatori”, “Adaugare articol”, “Adaugare categorie”
 - **Utilizatorii cu rolul Editor** pot accesa “Afisare articole”, “Adaugare articol”
 - **Utilizatorii cu rolul User** pot accesa “Afisare articole”

- ✚ Pentru stilizarea meniului, se pot adauga proprietati de CSS pe clasa **navbar** (.navbar), pe **link-uri** (.navbar a) si pe link-uri in momentul in care se face **hover** (.navbar a:hover)

- Sugestie de implementare:

```
.navbar {  
    background-color: #dadada;  
    border: none;  
    box-shadow: 0 6px 12px #dadada;  
}  
  
.navbar a {  
    color: #2c2c2c;  
}  
  
.navbar a:hover {  
    color: #5cb85c;  
}
```

- ✚ **Fonturile** din platforma – pentru alegerea fonturilor se poate utiliza <https://fonts.google.com/> Se cauta sau se aleg fonturile dorite (ex: Roboto, Montserrat, Permanent Marker). Pentru adaugare se apasa acel “plus” din dreptul fontului. In fereastra deschisa, in sectiunea @import, se copiaza secventa de cod din interiorul tagului <style> (fara tagul <style>) si se adauga in proiect, in Content -> Site.css -> pe prima linie. Dupa acest pas, se pot utiliza aceste fonturi cu ajutorul atributului **font-family**.

```
font-family: 'Roboto', sans-serif;  
font-family: 'Montserrat', sans-serif;  
font-family: 'Permanent Marker', cursive;
```

- ✚ **Stilizare logo** – pentru logo se poate utiliza fontul Permanent Marker si se poate stiliza cu ajutorul unei clase (in Site.css) dupa care se foloseste clasa in _Layout.cshtml (de exemplu:
`@Html.ActionLink("Articles App", "Index", "Article", new { area = "" }, new { @class = "navbar-brand font-logo" })`)

- ✚ **Culorile folosite** – pentru alegerea culorilor se poate utiliza <https://flatuicolors.com/>

- ✚ **Footerul** – in _Layout.cshtml, in tagul <footer> sa se modifice astfel incat numele ales pentru logo sa se afle si in aceasta sectiune. De asemenea, fontul ales pentru logo sa se utilizeze si in footer. Pentru includerea unui tag in interiorul altui tag se poate utiliza tagul . Footerul trebuie sa raman fix in partea de jos a paginii.

- ✚ **Pagina principala** – pe pagina principala /Home/Index sa se afiseze pentru **utilizatorii neautentificati** doar 3 articole, in functie de data publicarii. Pentru a verifica daca utilizatorul este autentificat se foloseste in metoda Index conditia `if (Request.IsAuthenticated)`. Daca este autentificat se va face redirect catre **metoda Index** din Controller-ul **Article** `RedirectToAction("Index", "Article")`. Daca utilizatorul nu este autentificat se preia primul articol folosind metoda `.First()`, iar

pentru a prelua urmatoarele doua articole se foloseste `Skip(1).Take(2)` deoarece nu trebuie sa afisam inca o data primul articol.

Primul articol sa se afiseze folosind clasa "jumbotron", (<https://getbootstrap.com/docs/3.3/components/#jumbotron>) folosind pentru titlu si continut fonturi diferite. De exemplu, pentru titlu fontul Roboto, iar pentru continut fontul Montserrat. Primul articol se va afisa pe un singur rand si pe o singura coloana, iar urmatoarele doua articole pe un singur rand, dar pe doua coloane. Pentru ultimele doua articole sa se afiseze si butoanele **Register** si **Log in**.

Se utilizeaza structura urmatoare pentru primul articol:

```
<div class="row">
    <div class="col-xs-12">
...
    </div>
</div>
```


Se utilizeaza structura urmatoare pentru urmatoarele doua articole:


```
<div class="row">
    @foreach (var article in ViewBag.Articles)
    {
        <div class="col-md-6 row-style">
            ...
        </div>
    }
</div>
```



Adaugati in fisierul Site.css, pentru clasa row-style spatiere in partea de sus de 50 px

De asemenea, pentru articolele de pe pagina principala sa nu se afiseze tot continut. Sa se afiseze doar 5 randuri din continutul fiecarui articol.

 **Paginare la afisarea tuturor articolelor** – pentru afisarea tuturor articolelor utilizam <https://getbootstrap.com/docs/3.3/components/#pagination> astfel incat sa afisam cate 3 articole pe pagina. Sa se modifice atat metoda Index din Controller-ul Articles, cat si View-ul Index corespunzator.

 **Editor text pentru adaugare si editare articol** – <https://summernote.org/getting-started/#installation>

In _Layout.cshtml -> in **Head** se include:

```
<link href="http://cdnjs.cloudflare.com/ajax/libs/summernote/0.8.12/summernote.css"
rel="stylesheet">
```

In **Footer**, inainte de inchiderea elementului <body> ->

```
<script src="http://cdnjs.cloudflare.com/ajax/libs/summernote/0.8.12/summernote.js">
</script>
```

In **Scripts** -> adaugam un nou fisier JavaScript care va contine urmatoarea secventa de cod:

```
$(document).ready(function () {
    $(''.summernote').summernote({
        height: 300,
        minHeight: 200,
        focus: true,
    });
});
```

Includem fisierul JavaScript in Layout, in Footer:

```
@Scripts.Render("~/Scripts/Summernote.js");
```

In View -> Articles -> New si Edit -> adaugam clasa **summernote** pe helper-ul **@Html.TextArea** astfel:

```
@Html.TextArea("Content", null, new { @class = "form-control summernote" }) ->  
in View-ul New
```

```
@Html.TextArea("Content", Model.Content, new { @class = "form-control  
summernote" }) -> in View-ul Edit
```

In **Controller-ul Articles** -> pe metodele New (POST) si Edit (PUT) se foloseste [**ValidateInput(false)**] deoarece daca acesta este activ nu se pot insera tag-uri html in baza de date.

In View pentru afisare, adica in partialul **ArticleInfo.cshtml** se foloseste **@Html.Raw** pentru afisarea si parsarea corecta a html-ului.

```
@Html.Raw(Model.Content)
```

Pe de alta parte, exista probleme de XSS deoarece acest helper nu face encoding la caractere. Acest lucru duce la posibilitatea inserarii unui cod JavaScript in continutul articolului, cod care va fi executat in pagina. Pentru a rezolva aceasta problema de securitate trebuie sa implementam in Controller, in metodele New (POST) si Edit (PUT):

New:

```
if (ModelState.IsValid)  
{  
    // Protect content from XSS  
    article.Content = Sanitizer.GetSafeHtmlFragment(article.Content);  
  
    db.Articles.Add(article);  
    db.SaveChanges();  
    TempData["message"] = "Articolul a fost adaugat";  
    return RedirectToAction("Index");  
}
```



Acest pachet se instaleaza folosind
NuGet Package Manager -> se cauta
AntiXSS

Edit:

```
if (TryUpdateModel(article))  
{  
    article.Title = requestArticle.Title;
```

```

        // Protect content from XSS
        requestArticle.Content =
Sanitizer.GetSafeHtmlFragment(requestArticle.Content);

        article.Content = requestArticle.Content;
        article.Date = requestArticle.Date;
        article.CategoryId = requestArticle.CategoryId;

        db.SaveChanges();
        TempData["message"] = "Articolul a fost modificat";
    }

```

Pentru mai multe detalii:

<http://www.dotnet-programming.com/post/2015/04/11/How-to-Handle-Cross-Site-Scripting-in-ASPNET-MVC-Application.aspx>

Sugestii de implementare:

✚ **Footerul** – sugestii de implementare:

- In _Layout.cshtml

```

<footer class="footer-position">
    <p>&copy; @DateTime.Now.Year - <span class="font-
        logo">Articles App</span></p>
</footer>

```

- In Site.css

```

.footer-position {
    position: fixed;
    left: 0;
    bottom: 0;
    width: 100%;
    text-align: center;
    padding-top: 15px;
    background: #fff;
    box-shadow: 0 6px 18px #ccc;
}

.font-logo {
    font-family: 'Permanent Marker', cursive;
}

```

Pagina principala /Home/Index – sugestii de implementare:

Home Controller – Metoda Index:

```
public ActionResult Index()
{
    if (Request.IsAuthenticated)
    {
        return RedirectToAction("Index", "Articles");
    }

    var articles = (from article in db.Articles
                    select article);

    ViewBag.FirstArticle = articles.First();
    ViewBag.Articles = articles.OrderBy(o => o.Date).Skip(1).Take(2);

    return View();
}
```

View-ul Index:

```
@{
    ViewBag.Title = "Home Page";
}

<div class="row">
    <div class="col-xs-12">
        <div class="jumbotron">
            <h1 class="font-roboto">@ViewBag.FirstArticle.Title</h1>
            <p class="truncate font-montserrat">
                @ViewBag.FirstArticle.Content
            </p>
        </div>
    </div>
</div>

<div class="row">
    @foreach (var article in ViewBag.Articles)
    {
        <div class="col-md-6 row-style">
            <h2 class="font-roboto">@article.Title</h2>
            <p class="truncate font-montserrat">
                @article.Content
            </p>

            <a class="btn btn-sm btn-success" href="/Account/Register">Register</a>
            <a class="btn btn-sm btn-success" href="/Account/Login">Log in</a>
        </div>
    }
</div>
```

Site.css

```
.truncate {
    /* Required declarations: */
    overflow: hidden;
    display: -webkit-box;
    -webkit-box-orient: vertical;
    /* Limit the text block to x lines */
    -webkit-line-clamp: 5;
}

.row-style{
    margin-top: 50px;
}
```

 **Paginare la afisarea tuturor articolelor – sugestii de implementare:**

Metoda Index

```
private ApplicationDbContext db = new ApplicationDbContext();

private int _perPage = 3;

// GET: Article
[Authorize(Roles = "User,Editor,Admin")]

public ActionResult Index()
{
    var articles =
db.Articles.Include("Category").Include("User").OrderBy(a => a.Date);

    var totalItems = articles.Count();

    var currentPage = Convert.ToInt32(Request.Params.Get("page"));

    var offset = 0;

    if (!currentPage.Equals(0))
    {
        offset = (currentPage - 1) * this._perPage;
    }

    var paginatedArticles = articles.Skip(offset).Take(this._perPage);
}
```



```

        if (TempData.ContainsKey("message"))
        {
            ViewBag.message = TempData["message"].ToString();
        }

        ViewBag.perPage = this._perPage;

        ViewBag.total = totalItems;

        ViewBag.lastPage = Math.Ceiling((float)totalItems /
            (float)this._perPage);

        ViewBag.Articles = paginatedArticles;

        return View();
    }

```

View-ul Index

```

<h2>News</h2>

<hr />

@if (TempData.ContainsKey("message"))
{
    <h3 class="alert alert-info" role="alert">@ViewBag.Message</h3>
}

@foreach (Laborator5App.Models.Article article in ViewBag.Articles)
{
    <div class="panel panel-article">
        @Html.Partial("ArticleInfo", article)
        <div class="panel-footer article-panel-footer around">
            <a class="btn btn-sm btn-success full-width"
href="/Articles/Show/@article.Id">Afisare articol</a>
        </div>
    </div>
    <br />
}

<div>
    <nav aria-label="Page navigation">
        <ul class="pagination">
            <li>
                <a href="/Articles/Index?page=1" aria-label="Previous">
                    <span aria-hidden="true">&laquo;</span>
                </a>
            </li>

```

```

        @for (int i = 1; i <= ViewBag.lastPage; i++)
        {
            <li><a href="/Articles/Index?page=@i">@(i)</a></li>
        }

        <li>
            <a href="/Articles/Index?page=@(ViewBag.lastPage)" aria-label="Next">
                <span aria-hidden="true">&raquo;</span>
            </a>
        </li>
    </ul>
</nav>
</div>

```

View-ul partial ArticleInfo.cshtml

```

<div class="panel-heading article-heading">
    <span class="font-roboto article-title">@Model.Title</span>
</div>

<div class="panel-body">
    <span class="article-content-title">Continut articol</span>
    <br />
    <div class="article-content-wrapper">
        @Html.Raw(Model.Content)
    </div>
</div>
<div class="article-meta">
    <span class="">
        <i class="glyphicon glyphicon-time"></i>
        @Model.Date
    </span>
    <div>
        <i class="glyphicon glyphicon-bookmark"></i>
        @Model.Category.CategoryName
    </div>
    <div>
        <i class="glyphicon glyphicon-user"></i>
        <strong> @Model.User.UserName </strong>
    </div>
</div>

```

```

/**
  Stilizare articol
*/

.panel-article {
  box-shadow: 0 2px 6px #ccc;
  border-radius: 0 !important;
}

.article-panel-footer.around {
  justify-content: space-around;
}

.article-heading {
  background-color: #fff !important;
  border-radius: 0 !important;
  border: none !important;
  box-shadow: 0 2px 6px #f2f2f2;
}

.article-title {
  font-size: 16px;
  font-weight: bold;
  color: #3c3c3c;
}

.article-content-title {
  font-weight: bold;
  color: #3c3c3c;
}

.article-content-wrapper {
  padding-top: 15px;
  padding-bottom: 15px;
}

.article-content-wrapper * {
  font-size: 13px !important;
}

.article-meta {
  display: flex;
  flex-direction: row;
  align-content: space-between;
  align-items: center;
  justify-content: space-between;
  padding: 15px;
}

```

```
.article-meta:hover{  
    background-color: #fafafa;  
}
```

```
.article-meta > * {  
    display: block;  
}
```

<https://caniuse.com/> - putem verifica ce proprietati de css pot fi utilizate in functie de browser