Dezvoltarea Aplicațiilor Web utilizând ASP.NET Core MVC
Laborator 4

EXERCIȚII:

Exercițiul 1:

Să se creeze un nou proiect, numit Laborator4, și să se adauge Controller-ul **ArticlesController**, în care se vor implementa următoarele (**VEZI** Curs 4 – capitolul Controller):

1. Să se adauge un Model numit **Article** care să conțină **Id**, **Title**, **Content** și **Date**, astfel: Models -> click dreapta -> Add -> Class -> Adăugăm o clasă Article.cs

```
public class Article
    {
        public int Id { get; set; }
        public string Title { get; set; }
        public string Content { get; set; }
        public DateTime Date { get; set; }
    }
```

2. Să se adauge în Controller o metodă **NonAction**, numita `GetArticles()` (**VEZI** Curs 4 – în secțiunea **Selectori**), care va returna un array de obiecte de tip Article, array pe care o să îl folosim pentru afișare (procedăm astfel deoarece în acest laborator nu o să folosim baza de date, iar în acest mod ne creăm articole pe care le putem prelucra).

CEZARA BENEGUI

```
(*) Cum se poate implementa o metodă NonAction? Ce reprezintă
o metodă NonAction?
```

```
public Article[] GetArticles()
 {
     // Se instantiaza un array de articole
     Article[] articles = new Article[3];

     // Se creeaza articolele
     for (int i = 0; i < 3; i++)
     {
         Article article = new Article();
         article.Id = i;

         article.Title = "Articol " + (i + 1).ToString();
         article.Content = "Continut articol " + (i + 1).ToString();
         article.Date = DateTime.Now;

         // Se adauga articolul in array
         articles[i] = article;
     }
     return articles;
 }
```

3. Să se adauge toate metodele pentru operațiile de tip C.R.U.D.

   - **Index** – pentru listarea tuturor articolelor;
   - **Show** – pentru vizualizarea unui articol în funcție de Id;
   - **New** – pentru crearea unui nou articol;
   - **Edit** – pentru editarea unui articol existent;
   - **Delete** – pentru ștergerea unui articol;

De asemenea, se vor adăuga verbele HTTP potrivite pentru fiecare metodă în parte (**VEZI** Curs 4 – secțiunea Selectori).

CEZARA BENEGUI

4. **Index**

- Să se creeze metoda Index în Controller-ul ArticlesController

- Să se creeze un **View** numit **Index**

În folderul View → se adaugă un folder nou corespunzător Controller-ului Articles astfel → Click dreapta pe folderul Views → Add → New Folder → Folderul o să se numească Articles. În acest folder se creează un fișier .cshtml asociat metodei Index → Click dreapta pe folderul Articles → Add → View → Se selectează opțiunea Razor View Empty → Add → Se selectează ASP.NET Core și Razor View – Empty → Se modifică denumirea în Index → Add

- În continuare se utilizează array-ul de articole, astfel incât să preluăm în metoda Index toate articolele, după care să le trimitem către View-ul asociat pentru afișare către utilizatorul final;

Metoda Index din ArticlesController:

```
public IActionResult Index()
    {
        Article[] articles = GetArticles();

        // Se adauga array-ul de articole in View
        ViewBag.Articles = articles;

        return View();

    }
```

Index.cshtml – în View-ul Articles

Să se implementeze afișarea articolelor. De asemenea, pentru fiecare articol trebuie să existe posibilitatea afișării și editării acestuia.

CEZARA BENEGUI

5. Modificați ruta **Default** din Program.cs, astfel încât la rularea proiectului să ne redirecționeze de fiecare dată către pagina de listare a tuturor articolelor.

6. **Show**

   - În metoda Show din ArticlesController afișați detaliile unui articol, în funcție de id-ul acestuia. În cazul în care articolul nu este găsit, afișați un View de eroare cu explicațiile aferente (mesajul de eroare aruncat de excepție și un mesaj – "Articolul căutat nu poate fi găsit").

   - Să se creeze un View numit Show, în care să se implementeze afișarea unui singur articol. View-ul va conține și un link către pagina de afișare a tuturor articolelor.

   - Configurați o rută pentru ruta existent, **/Articles/Show/{id}**, care după cum se observă se poate accesa prin intermediul rutei Default, astfel încât să se acceseze prin ruta **/articole/show/{id}**. Ruta o să se numească **ArticlesShow**

ArticlesController → Show

```
// Afisarea unui singur articol
public IactionResult Show(int? id)
{
    Article[] articles = GetArticles();

    try
    {
        ViewBag.Article = articles[(int)id];
        return View();
    }

    catch (Exception e)
    {
        ViewBag.ErrorMessage = e.Message;
        return View("Error");
    }

}
```

CEZARA BENEGUI

View → Show.cshtml

```
@{
    ViewBag.PageName = “Show”;
}

<h2>@ViewBag.PageName</h2>

<hr />

<h1>@ViewBag.Article.Title</h1>
<p>@ViewBag.Article.Content</p>
<small>@ViewBag.Article.Date</small>

<hr />
<br />

<a href=”/Articles/Index”>Afisare articole</a>
```

View → Error.cshtml

```
@{
    ViewBag.Msg = “Articolul cautat nu poate fi gasit!”;
}

<h2>@ViewBag.Msg</h2>
<br />
<p>Error message: @ViewBag.ErrorMessage</p>
```

Index.cshtml → ruta configurata

```
<a href=”/articole/show/@article.Id”>Afisare articol</a>
// Ruta configurata
```

CEZARA BENEGUI

7. **New**

- În Controller o să existe două metode numite **New**.
  Prima metodă este de tip [HttpGet], utilizându-se pentru afișarea formularului prin intermediul căruia se completează datele unui articol.
  A doua metodă este utilizată pentru trimiterea datelor corespunzătoare articolului către server, pentru adăugarea noului articol în baza de date. Pentru trimiterea datelor, întotdeauna se folosește verbul [HttpPost].

- O să existe, de asemenea, două View-uri.
  Un View pentru afișarea formularului de creare a unui articol.
  Al doilea View (căruia îi dăm un nume diferit – de exemplu: NewPostMethod), folosit în momentul în care datele pentru creare vor fi trimise către server. În acest caz, al doilea View o să afișeze doar un mesaj deoarece nu avem încă acces la o bază de date.

- Link-ul către pagina de creare a unui articol o să fie adăugat în Index.

ArticlesController → New ([HttpGet])

```
// GET: Afisarea formularului de creare a unui articol

[HttpGet]
public IActionResult New()
{
    return View();
}
```

CEZARA BENEGUI

ArticlesController → New ([HttpPost])

```
        // POST: Trimiterea datelor despre articol catre server
pentru adaugare in baza de date

        [HttpPost]
        public IActionResult New(Article article)
        {
            // … cod creare articol …

            return View("NewPostMethod");
        }
```

View → New.cshtml

```
<h2>Afisare formular de adaugare articol</h2>

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

    <button type="submit">Adauga articol</button>

</form>
```

View →NewPostMethod.cshtml

```
@{
    ViewBag.New = "Articolul a fost adaugat cu succes!";
}

<h2>
    @ViewBag.New
</h2>
```

CEZARA BENEGUI

8. **Edit**

- În Controller o să existe două metode numite **Edit**.
  Prima metodă este de tip [HttpGet] și se utilizează pentru afișarea formularului care o să conțină datele unui articol existent în baza de date. Formularul se afișează cu scopul modificării câmpurilor (o parte din ele sau chiar toate).
  A doua metodă este utilizată pentru trimiterea datelor corespunzătoare articolului către server, pentru editarea articolului în baza de date. Pentru trimiterea datelor întotdeauna se folosește verbul [HttpPost]. În cazul API-urilor se utilizează [HttpPut], dar în ASP.NET Core MVC se poate utiliza [HttpPost] atât pentru New, cât și pentru Edit și Delete). A doua metodă poate returna un View, la fel ca în exemplul anterior sau se poate redirecționa către pagina principală a aplicației, pagina în care se afișează lista tuturor articolelor.

- Și în acest caz o să existe două View-uri.
  Un View pentru afișarea articolului din baza de date, articol care urmează să fie editat.
  Al doilea View (căruia îi dam un nume diferit – de exemplu: EditMethod), este folosit în momentul în care datele pentru editare vor fi trimise către server. În acest caz, al doilea View o să afișeze doar un mesaj deoarece nu avem înca acces la o bază de date.

ArticlesController → Edit ([HttpGet])

```
// GET: Afisarea datelor unui articol pentru editare

[HttpGet]
public IactionResult Edit(int? id)
{
    ViewBag.Id = id;
    return View();
}
```

CEZARA BENEGUI

ArticlesController → Edit ([HttpPost])

```
        // POST: Trimiterea modificarilor facute catre server
pentru stocare in baza de date

        [HttpPost]
        public IactionResult Edit(Article article)
        {
            // … cod adaugare articol editat in baza de date

            //return Redirect(“/Articles/Index”);

            return View(“EditMethod”);
        }
```

View → Edit.cshtml

```
<br />

<p>Afisare formular de editare articol – in acest formular
de preiau datele curente ale articolului</p>

<form method=”post” action=”/Articles/Edit/@ViewBag.Id”>

    <button type=”submit”>Editeaza articol</button>

</form>
```

View → EditMethod.cshtml

```
@{
    ViewBag.New = “Articolul a fost editat cu succes!”;
 }

<h2> @ViewBag.New </h2>
```

CEZARA BENEGUI

9. **Delete**

   - În pagina Show inserați formularul corespunzător ștergerii intrării, prin **metoda DELETE**, folosind un buton la fel ca în exemplele anterioare. Verbul folosit este [HttpPost]. În cazul implementării unui API, verbul HTTP o să fie [HttpDelete]. În ASP.NET Core MVC se poate utiliza [HttpPost] atât pentru New, cât și pentru Edit și Delete.

ArticlesController → Delete ([HttpPost])

```
// POST Stergere articol din baza de date

[HttpPost]
public IactionResult Delete(int? id)
{
    // … cod stergere articol din baza de date

    return Content("Articolul a fost sters din baza de date!");
}
```

View → Show.cshtml

```
<form method="post" action = "/Articles/Delete/@ViewBag.Article.Id">
    <button type="submit">Stergere articol</button>
</form>
```

10. Redenumiți metoda Index din Controller în ***listare***, folosind selectori (**VEZI** Curs 4 – secțiunea Selectori). **Ce observați după redenumire? Ce modificări trebuie realizate?**

TEMĂ:

/!\ Finalizați toate exercițiile din acest laborator.

CEZARA BENEGUI