Dezvoltarea Aplicatiilor Web-Anul 3
Laborator 3

Exercitiul 1

Sa se adauge Controller-ul **ArticleController** in care se vor implementa urmatoarele:

- Sa se adauge un model numit Article care sa contina Id, Title, Content si Date astfel: Models -> click dreapta -> Add -> Class -> Adaugam o clasa 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; }
}
```

- Sa se adauge in Controller o metoda **NonAction** (**Vezi** Curs 3 – in sectiunea **Selectori**) care va returna un array de obiecte de tip Article, array pe care o sa il folosim pentru afisare (procedam astfel deoarece in acest laborator nu o sa folosim baza de date, iar in acest mod ne cream articole pe care le putem prelucra)

```
(*) Cum putem face aceasta metoda NonAction? Ce reprezinta o metoda Non Action?

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

    // Cream articole
    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;

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

- Sa se adauge toate metodele pentru operatiile de tip C.R.U.D. (**Index** – pentru listarea tuturor articolelor, **Show** – pentru vizualizarea unui articol, **New** – pentru crearea unui nou articol, **Edit** – pentru editarea unui articol, **Delete** – pentru stergerea unui articol). De asemenea, se vor adauga verbele HTTP potrivite pentru fiecare metoda in parte.

- Pentru **metoda Index** o sa procedam astfel: folosim array-ul de articole creat, pe care o sa il afisam si in View – **cream un View numit Index** (In folderul View, in folderul corespunzator Controller-ului nostru – Article – Click dreapta Article -> Add -> View -> Schimbam denumirea in Index -> Adaugam acest View creat).

  Metoda Index in Controller:

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

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

      return View();
}
```

Index.cshtml – in View

```
@{
    ViewBag.Title = "Index";
}

<h2>@ViewBag.Title</h2>

@foreach (var article in ViewBag.Articles)
{
    <h1>@article.Title</h1>
    <p>@article.Content</p>
    <small>@article.Date.ToString()</small>
    <a href="/Article/Show/@article.Id">Afisare articol</a>
    <a href="/Article/Edit/@article.Id">Editare articol</a>
    <br />
    <hr />
    <br />
}

<a href="/Article/New ">Adaugare articol</a>
```

- Modificati ruta Default din RouteConfig.cs astfel incat la rularea proiectului sa ne redirectioneze de fiecare data catre pagina de listare a tuturor articolelor.

- Creati o ruta pentru ruta existenta /Article/Show/{id} care dupa cum se oberva se poate accesa prin intermediul rutei Default, astfel incat sa se acceseze prin ruta /articles/show/{id}

- Redenumiti metoda Index din Controller in “listare”, folosind selectori (**Vezi** Curs 3 – sectiunea Selectori). Ce observati dupa redenumire? Ce modificari trebuie realizate?

- **Metoda Show** va afisa detaliile articolului. In cazul in care articolul nu este gasit, sa afiseze un View de eroare cu explicatiile aferente (mesajul de eroare aruncat de exceptie si un mesaj – “Articolul cautat nu poate fi gasit”). View-ul pentru Show va contine si un link catre pagina de afisare a tuturor articolelor.

- **Metoda New** o sa aiba doua View-uri asociate: unul pentru afisarea formularului de creare a unui articol (aceasta metoda va avea implicit metoda GET) si un View (caruia ii dam un nume diferit – de exemplu: NewPostMethod) folosit in momentul in care datele pentru creare vor fi trimise catre server. Pentru trimiterea datelor intotdeauna se foloseste metoda **POST** astfel:

```
<form method="post" action="/Article/New">
      <button type="submit">Adauga articol</button>
</form>
```

- In View-ul pentru metoda New trebuie sa existe un formular (tag-ul **form**) care contine metoda POST, la fel ca exemplul anterior

Exemplu pentru metoda New si View-urile asociate:

```
// GET: Afisarea formularul de crearea a unui articol
  public ActionResult New()
  {
      return View();
  }

  // POST: Trimiterea datelor despre articol catre server pentru creare
  [HttpPost]
  public ActionResult New(Article article)
  {
      // ... cod creare articol ...
      return View("NewPostMethod");
  }
```

New.cshtml

```
@{
    ViewBag.Title = "New";
}

<h2>Creare articol</h2>
<p>Afisare formular de adaugare articol</p>
<form method="post" action="/Article/New">
    <button type="submit">Adauga articol</button>
</form>
```

NewPostMethod.cshtml

```
@{
    ViewBag.Title = "NewPostMethod";
}

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

- Se procedeaza in acelasi mod si cu **metoda Edit** (**OBS:** Vezi Curs 3 – despre necesitatea folosirii metodei **HttpMethodOverride**)

- In pagina Show putem insera si formularul corespunzator stergerii intrarii prin **metoda DELETE**, folosind si un buton la fel ca in exemplele anterioare

Exercitiul 2:

Sa se creeze un folder numit "**ActionFilters**". Sa se implementeze o clasa numita LogFilter.cs care contine un filtru **OnActionExecuting** (**Vezi** Curs 3 – sectiunea Filtre – exemplul 2). Acest filtru va afisa un mesaj in consola IDE-ului care va contine urmatoarele:

- URL-ul accesat - **numeParam.HttpContext.Request.Url**

- Data la care a fost accesat URL-ul – **DateTime.Now**

- Adresa IP a utilizatorului - **numeParam.HttpContext.Request.UserHostAddress**

- Detalii despre browser-ul utilizatorului – **numeParam.HttpContext.Request.UserAgent**

Implementati clasa si metodele sale, iar apoi folositi acest filtru custom la nivelul actiunilor Index si Show in Controller-ul ArticleController.

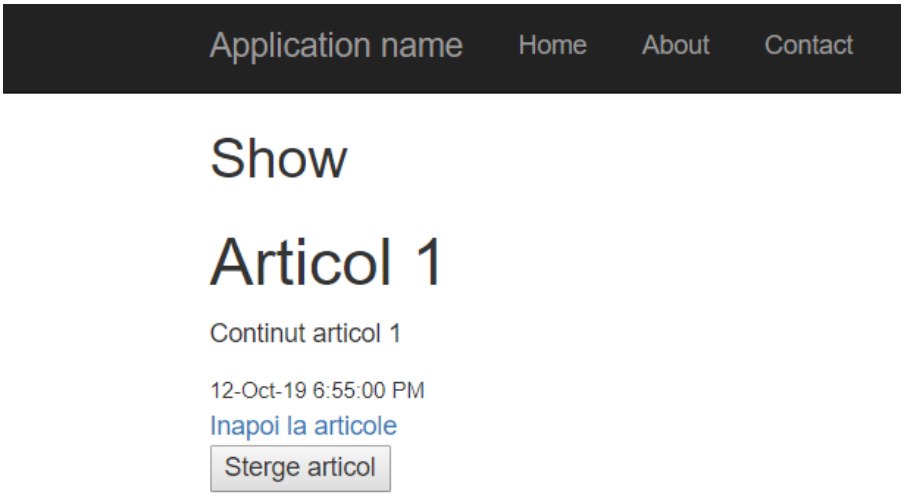
Exercitiul 3:

Sa se adauge pe actiunea Index din **ArticleController** un cache care dureaza 30 de secunde (**Vezi** Curs 3 – sectiunea Filtre – exemplul 1). Porniti aplicatia si incarcati acea pagina. Dupa incarcarea si vizualizarea paginii, modificati View-ul asociat acesteia, adaugand un mesaj. Observati ce se intampla in timpul celor doua minute si dupa scurgerea timpului.

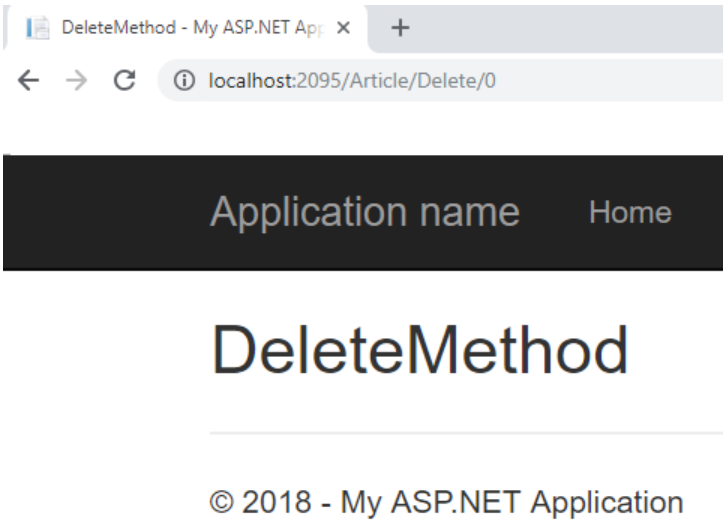
Exemplu pentru fiecare pagina creata:

Pagina Index:

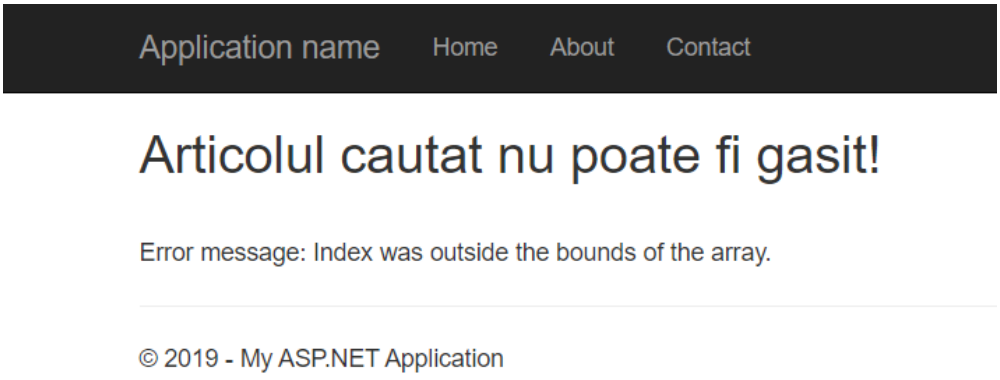
Pagina Show:



Dupa apasarea butonului “Sterge articol” din pagina **Show.cshtml** se va afisa pagina urmatoare:

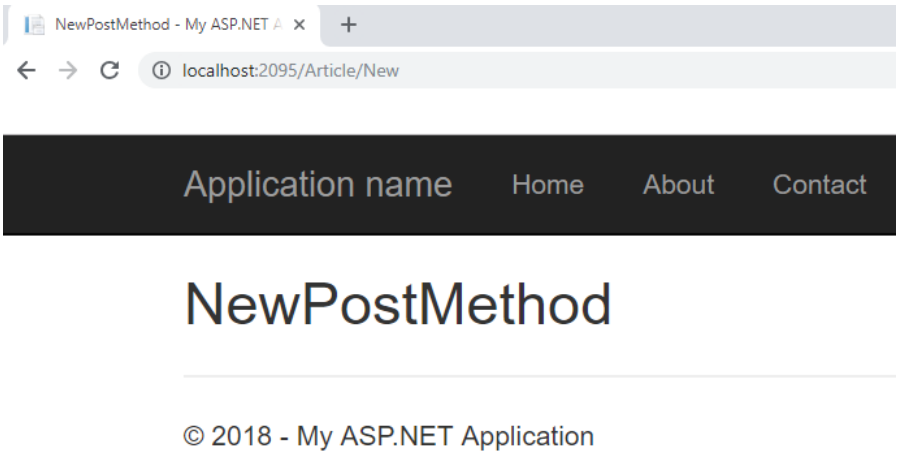


Daca se cauta un articol neexistent se va afisa urmatorul View de eroare:

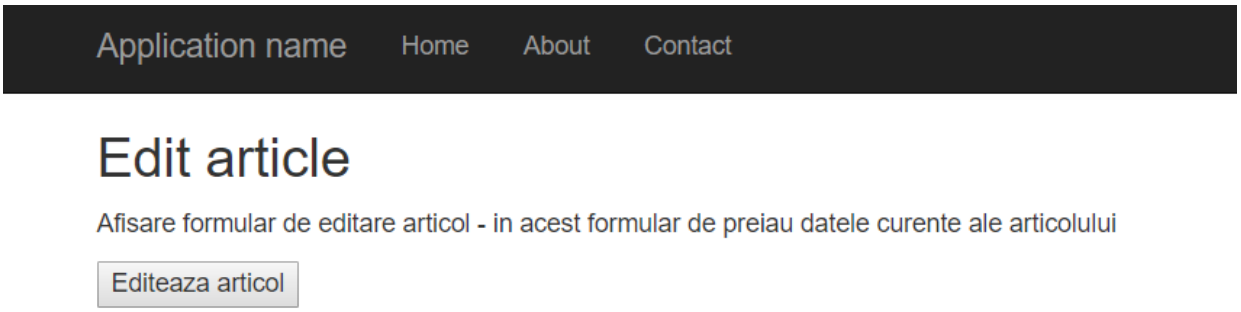


Pagina de creare articol (New)

Dupa apasarea butonului din imaginea anterioara (“**Adauga Articol**”) se va afisa pagina urmatoare:



Pagina de modificare articol (Edit)



Dupa apasarea butonului “**Editeaza articol**” se va afisa pagina: