

Web-aplikacije

ASP.NET Core MVC

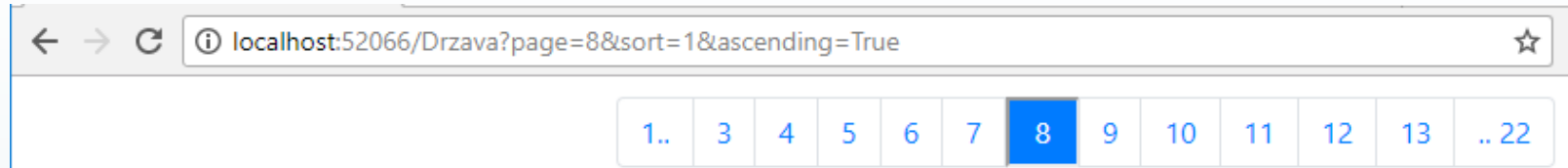
2017/18.09

Kreiranje vlastitog *tag-helpera* za straničenje

Kreiranje poveznica za straničenje

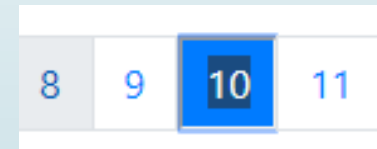
2

- Potrebno izraditi vlastitu komponentu/kontrolu koja će omogućiti prikaz poveznica za odlazak na pojedinu stranicu



- Željena funkcionalnost

- Trenutnu stranicu prikazati drugačijim stilom
- Prikazati poveznice na prethodnih n i na sljedećih n stranica
 - n je parametar u konfiguracijskoj datoteci
- U slučaju da je prva, odnosno zadnja stranica udaljena za više od n od trenutne stranice prikazati i poveznicu na prvu, odnosno zadnju stranicu
- Omogućiti unos broja željene stranice
 - npr. klikom na trenutnu stranicu i upisom broja
 - Izvršiti provjeru unesenog broja prije preusmjerenja
- Komponenta mora raditi s bilo kojim entitetom iz primjera, a ne samo s državama i treba voditi evidenciju o trenutnom načinu sortiranja
 - Koristi se informacija iz razreda *PagingInfo*



Podsjetnik na (neke) ugrađene *tag-helper*ere

3

- Želi li se stvoriti poveznica za akciju Show u upravljaču Contact gdje je vrijednost parametra *name* jednaka Pero Perić ugrađeni tag helperi će se primijeniti na standardnu HTML-ovu oznaku *a*


```
<a asp-controller="Contact" asp-action="Show"  
    asp-route-name="Pero Perić" ...>Pero Perić</a>
```

- Atributi *asp-controller*, *asp-action*, *asp-route-naziv* i slično se interpretiraju prilikom iscrtavanja pojedinog pogleda
- Komponenta za straničenje bit će izvedena pomoću vlastitog *tag-helper*a s vlastitim atributima čija se vrijednosti dohvaćaju u kodu *tag-helper*a i uzrokuju generiranje potrebnih poveznica ispod nekog *div* elementa u HTML-u.

```
<pager vlastitiatribut1="vrijednost"  
        vlastitiatribut2="vrijednost" ... ></pager>
```

Izrada vlastitog *tag-helpera*

4

- Stvara se izvođenjem iz razreda *TagHelper*
- Atributom *HtmlTargetElement* navodi se na koju HTML oznaku se može primijeniti (ako ne na onu koja odgovara nazivu tag-helpera) te koji skup atributa je obvezan
 - Može se navesti skup obveznih atributa (odvajaju se zarezom)
- Argumenti konstruktora stvaraju se koristeći *DependencyInjection*
 - Primjer:  Web \ Firma.Mvc \ TagHelpers \ PagerTagHelper.cs


```
[HtmlTargetElement(Attributes = "page-info")]
public class PagerTagHelper : TagHelper {

    private readonly IUrlHelperFactory urlHelperFactory;
    private readonly AppSettings appData;

    public PagerTagHelper(IUrlHelperFactory helperFactory,
                        IOptionSnapshot<AppSettings> options) {
        urlHelperFactory = helperFactory;
        appData = options.Value;
    }
}
```

Uključivanje vlastitog *tag-helpera*


5

- Može se uključiti u pojedinom pogledu ili za sve pogleda navođenjem u *_ViewImports.cshtml*
- Primjer:  Web \ Firma.Mvc \ Views \ _ViewImports.cshtml

```
@using Firma.Mvc
@using Firma.Mvc.Models
@using Firma.Mvc.ViewModels
@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
@addTagHelper Firma.Mvc.TagHelpers.*, Firma.Mvc
```

Svojstva željenog *tag-helpera* (1)


6

- *Tag-helper* će se primjenjivati na HTML oznaku `div` i imat će sljedeće vlastite attribute
 - `page-info`: informacija o trenutnoj stranici i načinu sortiranja
 - `page-action`: akcija na koju poveznica vodi
 - `page-title`: tekst za tooltip za unos željene stranice
- Korišteni stilovi od Bootstrapa
 - Za polje za prikaz trenutne stranice i unos željene korišten vlastiti css stil `pagebox`
- Primjer korištenja:  ... Firma.Mvc \ Views \ Drzava \ Index.cshtml

```
<pager page-info="@Model.PagingInfo"  
        page-action="Index"  
        page-title="Unesite željenu stranicu"  
        class="float-right">  
</pager>
```

Svojstva željenog *tag-helpera* (2)

7

- Atributi *tag-helpera* navode se kao svojstva pri čemu se umjesto naziva s crticama koristi naziv u notaciji PascalCase
 - Za svojstva koja nisu predviđena za korištenje u pogledu iznad svojstva se stavlja atribut *HtmlAttributeNotBound* (npr. kontekst konkretne upotrebe)
- Primjer:  Web \ Firma.Mvc \ TagHelpers \ PagerTagHelper.cs

```
[HtmlTargetElement(Attributes = "page-info")]
public class PagerTagHelper : TagHelper {
    ...
    [ViewContext]
    [HtmlAttributeNotBound]
    public ViewContext ViewContext { get; set; }

    public PagingInfo PageInfo { get; set; }
    public string PageAction { get; set; }
    public string PageTitle { get; set; }
    ...
}
```

Rezultat *tag-helpera*

8

➤ Rezultat se priprema u nadjačanom postupku `Process`

- Kreira HTML na osnovi konteksta i vrijednosti svojstva. Umjesto direktnog stvaranja HTML-a koristi se razred *TagBuilder* za određenu HTML oznaku.
- Sadržaj ispisa s *output.Content.AppendHtml* se upisuje u HTML oznaku na kojoj je atribut primijenjen
 - U primjeru oznaku `pager` promijeni u `div`

➤ Primjer:  `Web \ Firma.Mvc \ TagHelpers \ PagerTagHelper.cs`


➤ Vlastiti postupak *BuildTagForPage* opisan na sljedećem slajdu

```
public override void Process(TagHelperContext context,
                             TagHelperOutput output) {

    int offset = appData.PageOffset;
    TagBuilder navTag = new TagBuilder("nav");
    TagBuilder paginationList = new TagBuilder("ul");
    paginationList.AddCssClass("pagination");
    navTag.InnerHtml.AppendHtml(paginationList);
    ... Poveznice za trenutnu stranicu i +,- offset
    ...
    output.TagName = "div";
    output.Content.AppendHtml(navtag);
}
```


Kreiranje poveznica *tag-helperom*


9

- Za kreiranje adrese koristi se objekt tipa *IUrlHelper* i anonimni razred s vrijednostima parametara
 - Može se dobiti na osnovi trenutnog konteksta i objekta tipa *IUrlHelperFactory*
 - Primjer:  Web \ Firma.Mvc \ TagHelpers \ PagerTagHelper.cs

```
private TagBuilder BuildTagForPage(int i, string text) {  
    IUrlHelper urlHelper =  
        urlHelperFactory.GetUrlHelper(ViewContext) ;  
    TagBuilder tag = new TagBuilder("a");  
    tag.InnerHtml.Append(text);  
    tag.Attributes["href"] = urlHelper.Action(PageAction,  
        new { page = i, sort = PageInfo.Sort,  
            ascending = PageInfo.Ascending });  
    tag.AddCssClass("page-link");  
  
    TagBuilder listItemTag = new TagBuilder("li");  
    listItemTag.AddCssClass("page-item");  
    listItemTag.InnerHtml.AppendHtml(tag);  
    return listItemTag;  
}
```

Kreiranje poveznice za trenutnu stranicu (1)


10

- Po uzoru na poveznicu za prvu stranicu kreiraju se i poveznice za stranice od $\max\{1, \text{trenutna} - n\}$ do $\min\{\text{ukupno}, \text{trenutna} + n\}$
 - Za trenutnu stranicu ne kreira se oznaka *a*, već tekstualni okvir
 - Postavljaju se dodatni atributi oblika *data-naziv* (vidi sljedeći slajd)
 - Stil *pagebox* služi da ga se *jQuery*jem može pronaći u dokumentu, a ujedno je stil definiran u stilskoj datoteci kako bi se odredila širina tog okvira
- Primjer:  Web \ Firma.Mvc \ TagHelpers \ PagerTagHelper.cs

```
IUrlHelper urlHelper = urlHelperFactory.GetUrlHelper(ViewContext);
TagBuilder tag = new TagBuilder("input");
tag.Attributes["type"] = "text";
tag.Attributes["value"] = text;
... Postavljanje dodatnih atributa ...
tag.AddCssClass("pagebox"); //za identifikaciju i širinu
TagBuilder listItemTag = new TagBuilder("li");
listItemTag.AddCssClass("page-item active");
listItemTag.InnerHtml.AppendHtml(tag);
return tag;
```

Kreiranje poveznice za trenutnu stranicu (2)


11

- Za provjeru ispravnosti korisnikovog unosa potrebno je evidentirati dozvoljeni raspon stranica
 - Evidentira se u atributima oblika `data-naziv` te se za dohvat vrijednosti koristi postupak `.data(naziv)` iz jQuerya
- Korisnik će unijeti željeni broj stranice i on se treba ugraditi u odredišnu adresu. Mjesto gdje bi trebao stajati broj stranice prepoznaje se po nekom specijalnom nizu znakova
 - U ovom primjeru -1, ali može biti i nešto drugo
- Primjer:  Web \ Firma.Mvc \ TagHelpers \ PagerTagHelper.cs

```
TagBuilder tag = new TagBuilder("input");  
...  
tag.Attributes["data-current"] = text;  
tag.Attributes["data-min"] = "1";  
tag.Attributes["data-max"] = PageInfo.TotalPages.ToString();  
tag.Attributes["data-url"] = urlHelper.Action(PageAction,  
    new { page = -1, sort = PageInfo.Sort,...});  
...
```

Označavanje sadržaja kontrole za unos stranice


12

- Klikom na kontrolu koja prikazuje trenutnu stranicu, a ujedno služi i za unos željene stranice njen sadržaj se automatski označi
 - Implementira se obradom događaja klika gumba
 - Na izvoru događaja poziva se postupak select()
 - Kontrolu prepoznamo po postojanju stila pagebox
- Povezivanje događaja i obrada događaja odvija se nakon što je cijeli dokument učitano
 - Konstrukcija \$(function() { ... }); u jQueryju
- Primjer:  Web \ Firma.Mvc \ wwwroot \ js \ gotopage.js

```
$(function () { //nakon što je dokument učitano
    $('.pagebox').click(function () {
        //obrada klika na sve kontrole koje imaju stil pagebox
        $(this).select(); //selektiraj sadržaj
    });
});
```

Akcije na tipke Enter i ESC u pregledniku


13

- Nakon svakog otpuštenog znaka na tipkovnici (događaj *keyup*) provjerava se radi li se o tipkama *enter* (kôd 13) ili *escape* (kôd 27)
 - Aktiviraju prijelaz na novu stranicu (prethodno zamijenivši -1 s valjanim upisanim brojem), odnosno vraćaju originalnu vrijednost
 - Primjer:  Web \ Firma.Mvc \ wwwroot \ js \ gotopage.js

```
$('.pagebox').bind('keyup', function (event) {  
    var keycode = (event.keyCode ? event.keyCode : event.which);  
    var pageBox = $(this);  
    if (keycode == 13) {  
        if (validRange(pageBox.val(), pageBox.data("min"), pageBox.data("max"))) {  
            var link = pageBox.data('url');  
            link = link.replace('-1', pageBox.val());  
            window.location = link;  
        }  
    }  
    else if (keycode == 27) {  
        pageBox.val(pageBox.data('current'));  
    }  
});
```

Provjera ispravnosti unosa broja stranice

14

- Provjera na klijentskoj strani korištenjem vlastite skriptne datoteke
 - Koristi se regularni izraz za provjeru radi li se o broju, a zatim se provjera je li željeni broj unutar raspona
- Primjer:  Web \ Firma.Mvc \ wwwroot \ js \ gotopage.js

```
function validRange(str, min, max) {  
    var intRegex = /^\\d+$/;  
    if (intRegex.test(str)) { //da li je upisan broj?  
        var num = parseInt(str);  
        if (num >= min && num <= max)  
            return true;  
        else  
            return false;  
    }  
    else {  
        return false;  
    }  
}
```