

Razvoj primijenjene programske potpore

6.2 Kreiranje vlastitog tag-helpera

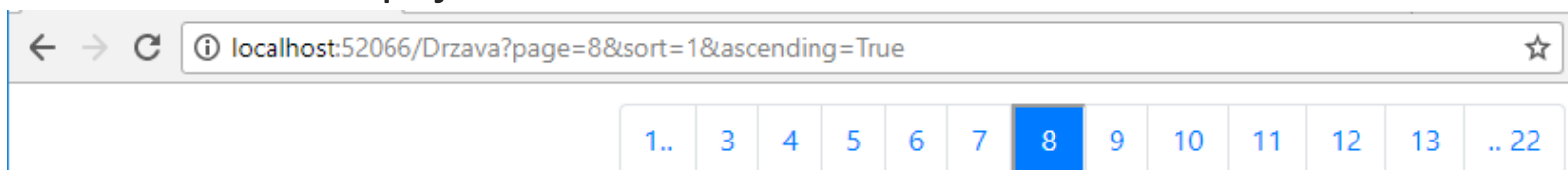
Dodatni materijali

Napomena vezana za primjer koji slijedi

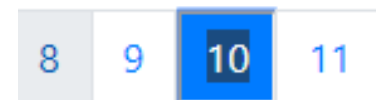
- Primjeri u ovim predavanjima predstavljaju složene koncepte koji uključuju korištenje CSS stilova i jQueryja
- Primjeri su opsežni i ne mogu u potpunosti stati na slajdove te se preporuča isprobati primjer i detaljno proučiti programski kôd pri čemu slajdovi mogu poslužiti kao vodilja kojim redom proučavati primjere

Kreiranje poveznica za straničenje

- 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 preusmjeravanja
 - 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*e


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

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

- 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 obavezan
 - Može se navesti skup obveznih atributa (odvajaju se zarezom)
- Argumenti konstruktora stvaraju se koristeći *DependencyInjection*
 - Primjer:  MVC \ TagHelpers \ PagerTagHelper.cs


```
[HtmlTargetElement(Attributes = "page-info")]
public class PagerTagHelper : TagHelper {
    private readonly IUrlHelperFactory urlHelperFactory;
    private readonly AppSettings appData;
    public PagerTagHelper(
        IUrlHelperFactory helperFactory,
        IOptionsSnapshot<AppSettings> options) {
        urlHelperFactory = helperFactory;
        appData = options.Value;
    } ...
```

Uključivanje vlastitog *tag-helpera*

- Može se uključiti u pojedinom pogledu ili za sve poglede navođenjem u *_ViewImports.cshtml*
- Primjer:  MVC \ Views \ *_ViewImports.cshtml*


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

Svojstva željenog *tag-helper* (1)

- *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:  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)

- 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:  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*

- 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 koji se ispiše pomoću *output.Content.AppendHtml* upisuje se u HTML oznaku na kojoj je atribut primijenjen
 - U primjeru oznaku *pager* promijeni u *nav*
- 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 paginationList = new TagBuilder("ul");
    paginationList.AddCssClass("pagination");
    navTag.InnerHtml.AppendHtml(paginationList);
    ... Poveznice za trenutnu stranicu i +,- offset ...
    output.TagName = "nav";
    output.Content.AppendHtml(paginationList);
}
```

Kreiranje poveznica *tag-helperom*

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


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

Kreiranje poveznice za trenutnu stranicu (1)

- 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 *jQueryjem* može pronaći u dokumentu, a ujedno je stil definiran u stilskoj datoteci kako bi se odredila širina tog okvira
- Primjer:  MVC \ TagHelpers \ PagerTagHelper.cs


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

Kreiranje poveznice za trenutnu stranicu (2)

- 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:  MVC \ TagHelpers \ PagerTagHelper.cs


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

Označavanje sadržaja kontrole za unos stranice

- 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 učitani
 - Konstrukcija `$(function() { ... });` u jQueryju
- Primjer:  MVC \ TagHelpers \ PagerTagHelper.cs


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

Akcije na tipke Enter i ESC u pregledniku

- 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:  MVC \ wwwroot \ js \ gotopage.js

```
$('.pagebox').keyup(function (event) {  
    var 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

- 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:  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;  
    }  
}
```