

# Web-aplikacije

## ASP.NET Core MVC









2020/21.Dodatni materijali.2

Parcijalni pogledi i dinamičko ažuriranje

# Ideja primjera

2

- Prilikom brisanja mjesta umjesto osvježavanja cijele stranice i ponovnog tabličnog prikaza samo ukloniti redak obrisano mjesta
- Omogućiti ažuriranje podataka o mjestima unutar tabličnog prikaza (bez odlaska na posebnu stranicu)

Poštanski broj	Naziv mjesta	Poštanski naziv mjesta	Država		
53234	Breštane	Udbina	Croatia		
<input type="text" value="33507"/>	<input type="text" value="Breštanovci"/>	<input type="text" value="Crnac"/>	<input type="text" value="Croatia"/>		
43252	Breza	Prgomelje	Croatia		
51213	Breza	Jurdani	Croatia		

# Akcija brisanja mjesta

3

➡ Postupak vraća Json anonimnog razreda s 2 svojstva

➡ Primjer:  MVC \ Controllers \ MjestoController.cs

➡ Ako mjesto ne postoji vraća se *NotFound*

```
[HttpPost][ValidateAntiForgeryToken]
public async Task<IActionResult> DeleteAjax (int id) {
    var mjesto = await ctx.Mjesto.FindAsync(id);
    if (mjesto != null) {
        try {
            string naziv = mjesto.NazMjesta;
            ctx.Remove(mjesto);
            await ctx.SaveChangesAsync();
            var result = new {
                message = $"Mjesto {naziv} sa šifrom {id}obrisano.",
                successful = true
            };
            return Json(result);
        }
    }
}
```

...

# Izvedba dinamičkog brisanja na klijentu (1)

4

- Na postojeći gumb za brisanje dodamo neki vlastiti atribut, npr. data-delete-ajax s adresom akcije koja treba obrisati element i vratiti Json.


➤ Primjer:  MVC \ Views \ Mjesto \ Index.cshtml

```
<form asp-action="Delete" method="post"...>
  <button class="btn btn-sm btn-danger delete"
    data-delete-ajax="@Url.Action("DeleteAjax",
                                new { id = mjesto.IdMjesta })"
    title="Obriši">
    ...
```

- Ovakva izvedba osigurava postojanje *AntiForgeryToken*a zbog elementa *form* te osigurava da klasično brisanje radi ako dinamičko brisanje nije implementirano.

## Izvedba dinamičkog brisanja na klijentu (2)

5

- Postojeći JavaScript kod izmijenjen tako da provjeri je li za *delete* gumb definiran atribut s prethodnog slajda
  - Ako da, poziva se vlastita metoda deleteAjax
  - Primjer:  MVC \ wwwroot \ js \ site.js

```
$(function () {  
    $(document).on('click', '.delete', function (event) {  
        if (!confirm("Obrisati zapis?")) {  
            event.preventDefault();  
        }  
        else {  
            const url = $(this).data('delete-ajax');  
            if (url !== undefined && url !== '') {  
                event.preventDefault();  
                deleteAjax($(this), url);  
            }  
        }  
    })  
    ...  
})
```

## Izvedba dinamičkog brisanja na klijentu (3)

6

➡ U slučaju uspješnog brisanja uklanja se redak u kojem se nalazio gumba za brisanje

➡ Primjer:  MVC \ wwwroot \ js \ site.js

```
function deleteAjax(btn, url) {  
    const tr = btn.parents("tr");//redak kojem zapis pripada  
    ...  
    const token = $('input[name="__RequestVerificationToken"]')  
        .first().val();  
    $.post(url, { __RequestVerificationToken:token},  
        function (data) {  
            if (data.successful) { //data je JSON koji server vrati  
                tr.remove();  
            }  
            ... Ispiši poruku o uspješnom brisanjeu ...  
        }).fail(function (jqXHR) {  
            alert(jqXHR.status + " : " + jqXHR.responseText);  
        })  
    ...  
}
```


# Ideja ažuriranja unutar retka

7

- Klikom na gumb za ažuriranje pozvati akciju na serveru koja vraća parcijalni pogled
- Postojeći redak s podacima sakriti i prikazati HTML iz parcijalnog pogleda
  - Ako korisnik odustane od ažuriranja, odbaciti redak za ažuriranje i otkriti skriveni redak
- Prilikom snimanja podataka pokupiti podatke iz retka, redak za ažuriranje obrisati i pozvati odgovarajuću akciju na serveru
  - Akcija na serveru vraća
    - Status 200 i parcijalni pogleda koji sadrži poslane podatke i validacijsku pogrešku
    - Status 204 (No Content) kojim se označava da je spremanje uspješno
      - Skriveni redak ukloniti i ponovo učitati podatke za taj redak
    - Nešto treće – prikazati poruku o pogrešci

# Priprema gumba za ažuriranje

8

- Na postojeći gumb za ažuriranje dodana 2 atributa koja sadrže
  - adresu akcije koja vraća parcijalni pogled s formom za ažuriranje
  - adresu akcije koja vraća pojedinačni redak s podacima o mjestu
- Primjer:  MVC \ Views \ Mjesto \ Index.cshtml

```
<a asp-action="Edit" ...  
  data-get-ajax="@Url.Action("GetAjax",  
                                new { id = mjesto.IdMjesta }) "  
  data-edit-ajax="@Url.Action("EditAjax",  
                                new { id = mjesto.IdMjesta }) "  
  title="Ažuriraj"...
```



# Akcija za prikaz pojedinačnog mjesta

9

► Primjer:  MVC \ Controllers \ MjestoController.cs

```
[HttpGet]
public async Task<IActionResult> GetAjax(int id) {
    var mjesto = await ctx.Mjesto
        .Where(m => m.IdMjesta == id)
        .Select(m => new MjestoViewModel {
            IdMjesta = m.IdMjesta,
            NazivMjesta = m.NazMjesta,
            PostBrojMjesta = m.PostBrMjesta,
            PostNazivMjesta = m.PostNazMjesta,
            NazivDrzave = m.OznDrzaveNavigation.NazDrzave
        })
        .SingleOrDefaultAsync();

    if (mjesto != null)
        return PartialView(mjesto);
    else
        return NotFound($"Neispravan id mjesta: {id}");
}
```

# Parcijalni pogled za prikaz pojedinačnog mjesta

10


➡ Izgledom mora odgovarati retku iz Index.cshtml

➡ Primjer:  MVC \ Views \ Mjesto \ GetAjax.cshtml

```
@model MjestoViewModel
<tr>
  <td class="text-left">@Model.PostBrojMjesta</td>
  ...
  <td>
    <a asp-action="Edit"   asp-route-id="Model.IdMjesta"
      class="btn btn-sm edit"
      data-get-ajax="@Url.Action("GetAjax",
                                new { id = Model.IdMjesta })"
      data-edit-ajax="@Url.Action("EditAjax",
                                new { id = Model.IdMjesta })"
      title="Ažuriraj">...
  </td>
  <td>
    <form asp-action="Delete" method="post" ...
```

# Akcija za prikaz retka za ažuriranje

11

- Izvedba slična kao kod „normalnog” ažuriranja
  - pripremiti države (sortiranje i straničenje se ne koristi)
  - Primjer:  MVC \ Controllers \ MjestoController.cs

```
[HttpGet]
public async Task<IActionResult> EditAjax(int id) {
    var mjesto = await ctx.Mjesto
                        .AsNoTracking()
                        .Where(m => m.IdMjesta == id)
                        .SingleOrDefaultAsync();

    if (mjesto != null) {
        await PrepareDropDownLists();
        return PartialView(mjesto);
    }
    else {
        return NotFound($"Neispravan id mjesta: {id}");
    }
}
```

# Parcijalni pogled za ažuriranje mjesta

12

- Svi elementi za unos označeni vlastitim atributom *data-save*, kao i gumbi za snimanje i odustajanje

➤ Primjer:  MVC \ Views \ Mjesto \ EditAjax.cshtml

```
@model Mjesto
<tr> <td class="text-left">
    <input asp-for="NazMjesta" class="form-control" data-save="" />
</td>
...
<td class="text-left">
    <select class="form-control" asp-for="OznDrzave"
        asp-items="ViewBag.Drzave" data-save=""> </select>
...
<input type="hidden" asp-for="IdMjesta" data-save="" />
    <button type="submit" class="btn btn-sm btn-primary save" ...
    <button class="btn btn-sm cancel" ...
...
<div asp-validation-summary="All"></div>
```

# Akcija za prihvatanje ažuriranih podataka

13

➡ Izvedba slična kao kod „normalnog” ažuriranja

➡ Primjer:  MVC \ Controllers \ MjestoController.cs

```
[HttpPost] [ValidateAntiForgeryToken]
public async Task<IActionResult> EditAjax(int id, Mjesto mjesto)
{
    ... provjera ispravnosti ...
    try {
        ctx.Update(mjesto);
        await ctx.SaveChangesAsync();
        return NoContent();
    }
    catch (Exception exc)
    {
        ModelState.AddModelError(...;
        await PrepareDropDownLists();
        return PartialView(mjesto);
    }
}
```

# Izvedba dinamičkog ažuriranja na klijentu (1)

14

➡ Za svaki *edit* gumb provjeriti ima li definiran adresu za prikaz retka za ažuriranje


➡ Ako da, poziva se vlastita metoda *editAjax*

➡ Primjer:  MVC \ wwwroot \ js \ site.js

```
$(document).on('click', '.edit', function (event) {  
    const editUrl = $(this).data('edit-ajax');  
    if (editUrl !== undefined && editUrl !== '') {  
        const getUrl = $(this).data('get-ajax');  
        event.preventDefault();  
        editAjax($(this), editUrl, getUrl);  
    }  
});
```

# Izvedba dinamičkog ažuriranja na klijentu (2)


15

- Vršiti se dohvat retka za ažuriranje (sa zastavicom pazimo na 2 brza klika – detaljnije u izvornom kodu)
  - Dohvaćeni HTML (tj. redak) dodajemo iza retka kojeg smo sakrili
- Definiramo ponašanje za *save* i *cancel* iz dohvaćenog retka
  - Primjer:  MVC \ wwwroot \ js \ site.js

```
function editAjax(btn, editUrl, getUrl) {  
  const tr = btn.parents("tr");//redak kojem zapis pripada  
  $.get(editUrl, { }, function (data) {  
    tr.toggle(); //sakrij trenutni redak  
    var inserted = $(data).insertAfter(tr);  
    setCancelAndSaveBehaviour(tr, inserted, editUrl, getUrl);  
  })  
  .fail(function (jqXHR) {  
    alert(jqXHR.status + " : " + jqXHR.responseText);  
  });  
  ...  
}
```

# Izvedba dinamičkog ažuriranja na klijentu (3)

16

- U slučaju odustajanja „redak” za ažuriranje odbacujemo, a vraćamo skriveni redak
  - Preciznije, „redak” za ažuriranje je HTML koji sadrži 2 retka zbog validacije i argument je *insertedData*
  - Primjer:  MVC \ wwwroot \ js \ site.js

```
function setCancelAndSaveBehaviour(  
    hiddenRow, insertedData, editUrl, getUrl) {  
  
    //nađi cancel button  
    insertedData.find(".cancel").click(function (event) {  
        insertedData.remove(); //ukloni umetnuti redak  
        hiddenRow.toggle(); //vрати vidljivost originalnom retku  
        hiddenRow.data('active', false);  
    });  
    ...  
}
```



# Izvedba dinamičkog ažuriranja na klijentu (4)

17

- ➡ Prilikom snimanja prikupimo podatke s forme (vlastita metoda *collectData*) i napravimo post zahtjev

➡ Primjer:  MVC \ wwwroot \ js \ site.js

```
...
insertedData.find(".save").click(function (event) {
    event.preventDefault();
    var formData = collectData(insertedData);

    $.ajax({
        type: "POST",
        url: editUrl,
        contentType: false,
        processData: false,
        data: formData,
        success: ...
        error: function (jqXHR) {
            alert(jqXHR.status + " : " + jqXHR.responseText);
        }
    });
});
```

# Izvedba dinamičkog ažuriranja na klijentu (5)

18

- Uspješan status može biti posljedica validacijske pogreške i prikaza HTML-a s pogreškom (status 200) ili uspješno snimanje (status 204)

➤ Primjer:  MVC \ wwwroot \ js \ site.js

```
...
success: function (data, textStatus, jqXHR) {
    insertedData.remove();
    if (jqXHR.status == 204) { //podatak ažuriran, osvježi
        $.get(getUrl, {}, function (refreshedRow) {
            $(hiddenRow).replaceWith(refreshedRow);
        })
    }
    else { //200
        var inserted = $(data).insertAfter(hiddenRow);
        setCancelAndSaveBehaviour(hiddenRow, inserted,
                                   editUrl, getUrl);
    }
}
```

# Izvedba dinamičkog ažuriranja na klijentu (6)

19

- Podatke s forme prikupimo tražeći sve elemente s atributom data-save, pazeći na interpretaciju (checkbox, file, ...)

➤ Primjer:  MVC \ wwwroot \ js \ site.js

```
function collectData(container) {  
    var formData = new FormData();  
    container.find("[data-save]").each(function (index, element) {  
        if ($(element).is(':checkbox'))  
            formData.append($(element).attr('name'),  
                            $(element).is(':checked'));  
  
        ...  
        else {  
            var val = $.trim($(element).val());  
            if (val != '')  
                formData.append($(element).attr('name'), val);  
        }  
    });  
    ... dodaj i antiforgery token ...  
    return formData;  
}
```