


Razvoj primijenjene programske potpore

7.1. Dodatni materijali Slanje datoteke i prikaz slike

Organizacija primjera

- Primjer izveden s dinamičkim ažuriranjem i brisanjem kao kod primjera s mjestima
 - Primjer:  MVC \ Views \ Artikl \ Index.cshtml

```
@model ArtikliViewModel
...
@foreach (var artikl in Model.Artikli) {
    <partial name="Get" model="artikl" />
}
```

- Prilikom ažuriranja artiklu se može pridružiti nova slika, obrisati postojeća ili ostaviti podatak o slici neizmijenjenim

Slike artikala

- Slika artikla pohranjena u tablici Artikl → Što sadrži model za prikaz pojedinog artikla
 - Preuzeti sliku zajedno s ostalim podacima o artiklu?
→ sporiji dohvat i korisnik više čeka na rezultat
 - U upitu dohvatiti samo osnovne podatke o artiklu, a sliku isporučiti na zahtjev?
- *Dilema neovisna o načinu dohvata podataka:*
 - *Gdje spremati slike?*
 - *Serverski cache za slike?*

Problem promjene slike artikla

- Dohvat slike se vrši po šifri artikla, a ne po identifikatoru slike
- Što ako preglednik sliku pospremi u cache, a slika se promijeni?
- Varijanta 1:
 - Nova slika = nova adresa!
- Kako na serveru znati da je slika nova?
 - Svakoj slici pridijeliti jedinstveni broj?
 - Koristiti neku varijantu sažetka?
- U tablicu Artikal je dodano izračunato polje *CHECKSUM(SlikaArtikla)*

RPPP2\SQL2012.Firma - dbo.Artikal

	Column Name	Data Type	Allow Nulls
🔑	SifArtikla	int	<input type="checkbox"/>
	NazArtikla	nvarchar(255)	<input type="checkbox"/>
	JedMjere	nvarchar(5)	<input type="checkbox"/>
	CijArtikla	money	<input type="checkbox"/>
	ZastUsluga	bit	<input type="checkbox"/>
	TekstArtikla	nvarchar(MAX)	<input checked="" type="checkbox"/>
	SlikaArtikla	varbinary(MAX)	<input checked="" type="checkbox"/>
▶	SlikaChecksum		<input checked="" type="checkbox"/>

Column Properties



Computed Column Specification

(checksum([SlikaArtikla]))

(Formula)

(checksum([SlikaArtikla]))

Is Persisted

No

Model za prikaz artikala

- Primjer:  MVC \ ViewModels \ ArtikliViewModel.cs


```
public class ArtikliViewModel
{
    public IEnumerable<ArtiklViewModel> Artikli { get; set; }
    public PagingInfo PagingInfo { get; set; }
}
```

- Model za pojedinačni artikal je razred *ArtiklViewModel*

- Primjer:  MVC \ ViewModels \ ArtiklViewModel.cs


```
public class ArtiklViewModel {
    public int SifraArtikla { get; set; }
    public string NazivArtikla { get; set; }
    public string JedinicaMjere { get; set; }
    public decimal CijenaArtikla { get; set; }
    public bool Usluga { get; set; }
    public string TekstArtikla { get; set; }
    public bool ImaSliku { get; set; }
    public int? ImageHash { get; set; }
}
```

Dohvat svih artikala

- Projekcija na prezentacijski model bez dohvata slike
 - Umjesto (sadržaja) slike, evidentira se postoji li slika
- Primjer:  MVC \ Controllers \ ArtiklController.cshtml


```
var artikli = ctx.Artikl
    .Select(a => new ArtiklViewModel {
        SifraArtikla = a.SifArtikla,
        NazivArtikla = a.NazArtikla,
        JedinicaMjere = a.JedMjere,
        CijenaArtikla = a.CijArtikla,
        Usluga = a.ZastUsluga,
        TekstArtikla = a.TekstArtikla,
        ImaSliku = a.SlikaArtikla != null,
        ImageHash = a.SlikaChecksum})
    .Skip(...)
    .Take(...)
    ...
```

Poveznica za prikaz slike

- Ako artikl koji se prikazuje u pojedinom retku ima sliku, stvara se HTML *img* kontrola
- Adresa slike je akcije *GetImage* na upravljaču *Artikl*
 - Adresi slike se dodaje parameter *hash* kako bi bili sigurni da preglednik zna da se radi o novoj slici
- Primjer:  MVC \ Views \ Artikl \ Get.cshtml


```
@if (Model.ImaSliku) {  
      
}
```

Akcija za dohvat slike

- Polje bajtova koje predstavlja sliku artikla dohvati se EF upitom
 - Rezultat postupka je *ActionResult*,
 - u slučaju da slika postoji vraća se binarni sadržaj pozivom postupka *File* iz upravljača.
 - Ako slike nema, vraća se status 404 s *NotFound*
 - Primjer:  MVC \ Controllers \ ArtiklController.cs


```
public async Task<ActionResult> GetImage(int id) {  
    ...  
    byte[] image = await ctx.Artikl  
                                .Where(a => a.SifArtikla == id)  
                                .Select(a => a.SlikaArtikla)  
                                .SingleOrDefaultAsync();  
    return File(image, "image/jpeg" ...  
    ...  
}
```


Kako sugerirati pregledniku da spremi sliku u cache? (1)

- Druga (i bolja) varijanta rješenja (problema) *cacheiranja* slika u pregledniku
- Koristi se *entity tag* (ETag)
 - Proizvoljni identifikator resursa kojim preglednik utvrđuje ima li aktualnu verziju resursa. Ako postoji preglednik ga šalje prilikom zahtjeva u zaglavlju *If-None-Match*
 - U slučaju da se tagovi poklapaju, vraća se status 304 (Not Modified)
- Primjer:  MVC \ Controllers \ ArtiklController.cs

```
public async Task<IActionResult> GetImage(int id) {  
    int? checksum = await ctx.Artikl  
        .Where(a => a.SifArtikla == id)  
        .Select(a => a.SlikaChecksum)  
        .SingleOrDefaultAsync();  
    if (checksum == null) return NotFound(); //slika ne postoji  
    string responseETag = "\"" + checksum.Value + "\"";  
    if (Request.Headers.TryGetValue(HeaderNames.IfNoneMatch,  
        out var requestETag) && requestETag == responseETag) {  
        return StatusCode((int)HttpStatusCode.NotModified);  
    }  
    ...  
}
```

Kako sugerirati pregledniku da spremi sliku u cache? (2)

- Kako je preglednik dobio ETag?
 - U primjeru se to dogodilo korištenjem jedne od varijanti postupka File što postavlja zaglavlja ETag u odgovoru
- Detaljnije na <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/If-None-Match>
- Primjer:  MVC \ Controllers \ ArtiklController.cs

▶ GET https://localhost:44395/Artikl/GetImage/4409?hash=876721973


Status	200 OK ?
Version	HTTP/2
Transferred	24.02 KB (23.79 KB size)

▼ Response Headers (239 B)

?	content-length: 24360
?	content-type: image/jpeg
?	date: Thu, 16 Dec 2021 21:37:40 GMT
?	etag: "876721973"
?	last-modified: Thu, 16 Dec 2021 21:37:40 GMT


```
public async Task<IActionResult> GetImage(int id) {  
    int? checksum = await ctx.Artikl  
        .Where(a => a.SifArtikla == id)  
        .Select(a => a.SlikaChecksum)  
        .SingleOrDefaultAsync();  
  
    ...  
    return File(image, "image/jpeg",  
        lastModified: DateTime.Now,  
        entityTag: new EntityTagHeaderValue(responseETag));  
}
```

Forma za slanje datoteke

- Prilikom unosa novog artikla može se poslati datoteka sa slikom
- Za unos se koristi HTML *input* kontrola tipa *file*
 - Naziv proizvoljan, ali mora odgovarati argumentu u akciji upravljača
- Forma mora imati atribut *enctype* postavljen na *multipart/form-data*
 - Primjer:  MVC \ Views \ Artikl \ Create.cshtml


```
<form asp-action="Create" method="post"
      enctype="multipart/form-data">
...
<label asp-for="SlikaArtikla" ...></label>
    <div class="col-sm-5">
        <input type="file" name="slika" />
    </div>
...
<button ... type="submit">Dodaj</button>
```

Prihvat datoteke na upravljaču

- Postupak prima objekt tipa *Artikl* (rekonstruiran na osnovi podataka iz forme) i objekt tipa *IFormFile*
 - Naziv argumenta jednak atributu *name* u kontroli za odabir slike
 - Sadržaj poslane podatke se može kopirati u *MemoryStream* i dobiti kao polje bajtova i pospremiti u entitet *Artikl*
 - U primjeru se originalna slika smanji prije pohrane u bazu podataka
- Primjer:  MVC \ Controllers \ ArtiklController.cs

```
[HttpPost][ValidateAntiForgeryToken]
... async Task<IActionResult> Create(Artikl artikl, IFormFile slika){
    ...
    if (slika != null && slika.Length > 0) {
        using (MemoryStream stream = new MemoryStream()) {
            await slika.CopyToAsync(stream);
            byte[] image = stream.ToArray();
            artikl.SlikaArtikla = image; //smanji prije pridruživanja
        }
    }
    ctx.Add(artikl);await ctx.SaveChangesAsync();    ...
}
```

Udaljena validacija na klijentskoj strani

- Provjerava postoji li već artikl s navedenom šifrom
 - sprječava se postavljanje upita koji će sigurno biti neuspješan
- Kako bi se korisniku ta informacija pružila i prije slanja forme koristi se tzv. udaljena validacija
 - Generira se javascript kod za poziv postupka na serveru (true/false)
 - Primjer:  MVC \ Models \ Artikl.cs

```
public partial class Artikl {  
    [Required]  
    [Remote(action: nameof(ArtiklController.ProvjeriSifruArtikla),  
            controller: "Artikl", ErrorMessage = "Šifra već postoji")]  
    public int SifArtikla { get; set; }  
}
```


- Primjer:  MVC \ Controllers \ ArtiklController.cs

```
public async Task<bool> ProvjeriSifruArtikla(int SifArtikla) {  
    return !await ctx.Artikl.AnyAsync(a => a.SifArtikla==SifArtikla);  
}
```

Napomena uz udaljenu validaciju

- Za razliku od ostalih atributa poput [Required], [Range] i slično, udaljena validacija se izvodi samo na klijentskoj strani
 - U slučaju da javascript kod nije ispravno izveden, model će se na serveru smatrati valjanim (tj. postupci za udaljenu validaciju neće biti pozvani)
- Moguće je napisati i vlastite validacijske attribute
 - Moguće dodati i vlastiti javascript kod za validaciju na klijentu
 - Više na: <https://docs.microsoft.com/en-us/aspnet/core/mvc/models/validation?#custom-attributes>

Ažuriranje artikla


- Ažuriranje osim podataka o artiklu prima novu sliku (ako postoji) i informaciju treba li možda obrisati sliku
 - Ne može se koristiti varijanta `ctx.Update(artikl)`, jer slika nije prenesena u pogled → dohvatiti artikl iz b.p. i ažurirati potrebno
 - Primjer:  MVC \ Controllers \ ArtiklController.cs

```
public async Task<IActionResult> Edit(Artikl artikl,
                                     IFormFile slika, bool obrisiSliku) {
    Artikl dbArtikl = await ctx.Artikl.FindAsync(artikl.SifArtikla);
    ...
    dbArtikl.JedMjere = artikl.JedMjere;
    ...
    if (slika != null && slika.Length > 0) {
        ... izvuci byte[] image iz streama
        dbArtikl.SlikaArtikla = image;
    }
    else if (obrisiSliku) dbArtikl.SlikaArtikla = null;
    await ctx.SaveChangesAsync;
```

Alati za smanjivanje slike

- Ugrađena podrška u CoreCompact.System.Drawing
- Neki od paketa za rad sa slikama
 - <https://andrewlock.net/using-imagesharp-to-resize-images-in-asp-net-core-a-comparison-with-corecompat-system-drawing>
 - <https://devblogs.microsoft.com/dotnet/net-core-image-processing>
- LibVips kao jedan od najbržih alata
 - <https://libvips.github.io/libvips/>
 - Od 4. mj. 2018. NetVips kao premostnica za .NET
 - https://kleisauke.github.io/net-vips/tutorial/getting_started.html

Uključivanje paketa NetVips

- Uključiti odgovarajuću verziju NetVipsa
- Ako na odredišnom računalu libvips nije instaliran i nisu postavljene varijable okruženja, može se uključiti u implementaciju
 - Potrebno uključiti za željenu platformu (npr. 64 bitne Windowse)
 - Primjer:  MVC \ MVC.csproj

```
<PackageReference Include="NetVips" Version="2.2.0" />  
<PackageReference Include="NetVips.Native.win-x64" Version="8.13.2" />
```

Dodatne aplikacijske postavke


- Konfiguracijska datoteka proširena s visinom smanjene slike
 - Primjer:  MVC \ AppSettings.cs

```
public class AppSettings {  
    ...  
    public ImageSettingsData ImageSettings { get; set; }  
    public class ImageSettingsData {  
        public int ThumbnailHeight { get; set; } = 100;  
    }  
}
```

- Primjer:  MVC \ appsettings.json

```
{  
  "AppSettings": {  
    "PageSize": 10, ...  
    "ImageSettings": {  
      "ThumbnailHeight": 100,  
    } ...  
  }  
}
```

Izrada smanjene slike

- Primjer:  MVC \ Util \ ImageUtil.cs
 - Postavimo ciljanu visinu ili širinu, a NetVips sačuva omjer
 - Druga dimenzija postavljena na neku vrijednost koja je veća od stvarne

```
const int VIPS_MAX_COORD = 10000000;
public static byte[] CreateThumbnail(byte[] image,
                                     int? maxwidth = null, int? maxheight = null) {
    if (maxwidth == null && maxheight == null)
        throw new ArgumentException(
            "Maximum size for at least one of axis must be specified");
    using (var thumbnailImage = Image.ThumbnailBuffer(image,
                                                       maxwidth ?? VIPS_MAX_COORD,
                                                       height: maxheight ?? VIPS_MAX_COORD)) {
        byte[] thumbnail = thumbnailImage.WriteToBuffer(".jpg");
        return thumbnail;
    }
}
```