.NET Okruženje

Vježba 9: Javascript, jQuery, date picker

Sadržaj

Javascript

Javascript – skriptni jezik koji se izvodi u svim web preglednicima, neophodan za izradu atraktivnih i funkcionalnih web aplikacija. Zadnjih nekoliko godina se koristi najčešće u kombinaciji s jQuery dodatkom za efikasno i jednostavno manipuliranje podacima i elementima. Vrlo je sličan C programskom jeziku, iako dopušta veliku razinu fleksibilnosti po pitanju tipova varijabli i samog prevođenja, što ga s druge strane čini malo kompliciranijim i težim za pronalazak pogrešaka.

Ukoliko želimo da se neki javascript kod izvede prilikom iscrtavanja HTML stranice u internet pregledniku, tada takav kod stavljamo u poseban HTML tag – script. Unutar script elementa mogu se nalaziti i funkcije koje neće odmah biti izvedene, već tek kad se pozovu. Također, postoji javascript kod koji se poziva na neku akciju koju korisnik izvodi – primjerice klik gumba ili prelazak mišem preko nekog određenog HTML elementa i sl. Gore navedeni primjeri su prikazani u donjem isječku koda:

Home/Index.cshtml

```
<h3>We suggest the following:</h3>
<ol class="round">
    <li class="one">
        ...
        <a href="http://go.microsoft.com/f       LinkId=245151">Learn more…</a>
    </li>
    <li class="two">
        ...
        <a href="http://go.microsoft.com/ wlink/?LinkId=245153">Learn more…</a>
    </li>
    <li class="three" onclick="onLiElementClick()">
        ...
        <a href="http://go.microsoft.com/fwlink/?LinkId=245157">Learn more…</a>
    </li>
</ol>

<script type="text/javascript">
    function onLiElementClick() {
        alert('<li> element clicked!');
    }

    alert('Kod koji se odmah izvodi!');
</script>
```

Kod koji se izvodi na klik tog elementa

Funkcija koja se izvodi tek na poziv

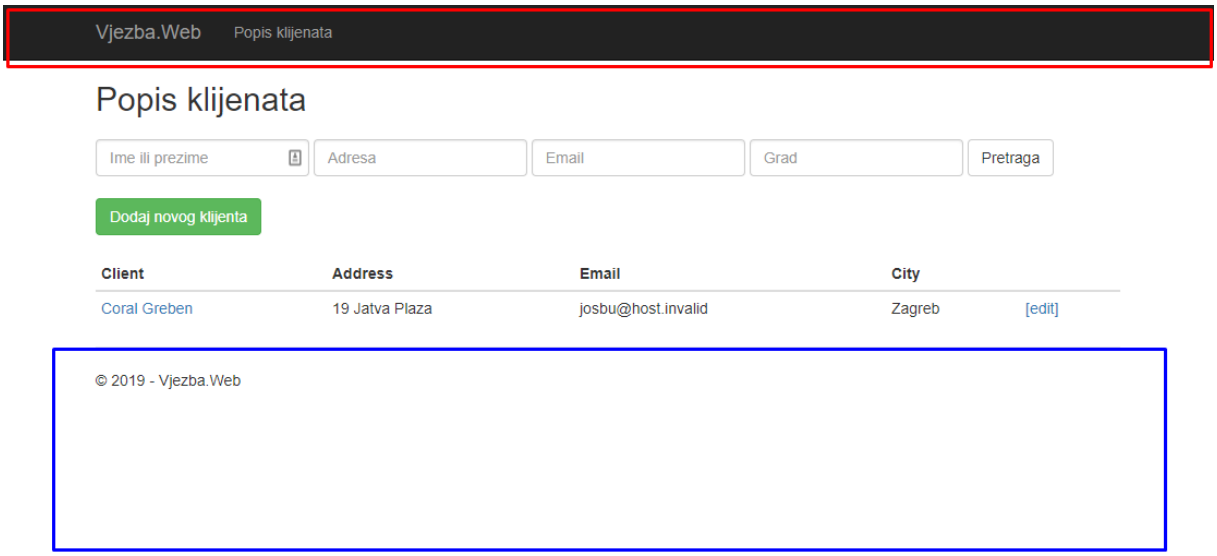
Kod koji se izvodi pri iscrtavanju stranice

Rukovanje HTML elementima u samom javascriptu je prilično nezgrapno, stoga je preporučljivo koristiti jQuery za manipulaciju elementima i podacima.

Redoslijed izvođenja JS koda i organizacija view datoteka

Zbog toga što browser izvodi kod stranice odozgo prema dolje (HTML/JS kod koji se nalazi prije će biti prije izveden), nije svejedno gdje stoji <script> tag unutar kojeg se nalazi JS kod.

Također, treba uzeti u obzir način kako se definira stranica u samom MVC radnom okviru. Na slici dolje prikazani su elementi sučelja s napomenom gdje se nalazi HTML kod koji definira izgled pojedinog elementa:



- Crveno uokvireno: header dio stranice, nalazi se u _Layout.cshtml
- Plavo uokvireno: footer stranice, nalazi se u _Layout.cshtml
- Ostalo (popis klijenata): nalazi se u Client/Index.cshtml

Layout pogled funkcionira na način da služi za iscrtavanje zajedničkih dijelova sučelja koji su onda korišteni na ostalim stranicama – najčešće je to izbornik i footer, obavijest o korištenju kolačića itd. Slijedi skraćeni dio _Layout stranice uz objašnjenje bitnijih dijelova koda:

1. Mjesto za meta podatke za aplikaciju: no što obično ide u <head> dio. Primjerice, <title> tag se generira iz ViewData dictionary-a, a upravo sadržaj <title> taga se puni u svakom view-u posebno.
2. Uključivanje CSS datoteka. Od .NET Core verzije, moguće je definirati različite biblioteke ovisno o tome gdje se nalazi aplikacija (<environment> tag). Najčešće će se različite datoteke uključiti ovisno o tome radi li se o *dev* ili produkcijskom/testnom okruženju. Ovaj element nije obvezan za koristiti.
3. Navigacija – za više informacija pogledati link: https://getbootstrap.com/docs/3.3/components/#navbar
4. Poziv partial view-a za prihvaćanje kolačića (cookie consent)
5. Ključni poziv – **RenderBody**. Ovaj dio je jedini obvezan unutar _Layout datoteke: na to mjesto se ubacuje sadržaj view-a koji gledamo. Primjerice, ako je pozvana akcija **Client/Edit**, tada se unutar _Layout stranice u pozivu RenderBody iscrtava view **Views/Client/Edit.cshtml**. Analogno vrijedi za bilo koji drugi view.
6. Footer - zajednička komponenta vidljiva na svakoj stranici
7. Uključivanje JS biblioteka – preporučljivo uključiti na kraju dokumenta kao u ovom primjeru, ali može se uključiti i u <head> dijelu.
8. Bitni dio: **@section**. S obzirom da se unutar RenderBody iscrtava svaki view posebno, nekad je potrebno s obzirom na neki view ubaciti elemente u ostale dijelove stranice – recimo u <head>, <navbar>, <footer> ili u ovom slučaju, omogućiti dodavanje javascript koda koji će se izvršavati **nakon** što su učitane sve JS biblioteke navedene iznad. Ovo je najčešće korišteni „section“, ali moguće je dodati i vlastite „section“ dijelove, gdje neki mogu biti obvezni a neki opcionalni.

_Layout.cshtml

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial 1 1.0" />
    <title>@ViewData["Title"] - Vjezba.Web</title>

    <environment include="Development">
        <link rel="stylesheet" href="~/css/site.css" />
    </environment>
    <environment exclude="Development">
        <link rel="stylesheet" href="~/css/site.min.css" />
    </environment>
</head>
<body>
    <nav class="navbar navbar-inverse navbar-fixed-top">
        ...
    </nav>

    <partial name="_CookieConsentPartial" />

    <div class="conta  dy-content">
        @RenderBody()
        <hr />
        <footer>
            <p>&copy; 2019 - Vjezba.Web</p>
        </footer>
    </div>

    <environment include="Development">
        <script src=https://ajax.aspnetcdn.com/ajax/jquery/jquery-3.3.1.min.js>
        </script>
        ...
        <script src="~/js/site.js" asp-ap  rsion="true"></script>
    </environment>
    <environment exclude="Development">
        <script src=https://ajax.aspnetcdn.com/ajax/jquery/jquery-3.3.1.min.js>
        </script>
        ...
        <script src="~/js/site.min.js" asp-append-version="true"></script>
    </environment>

    @RenderSection("Scripts", required: false)
</body>
</html>
```

1 2 3 4 5 6 7 8

Client/Index.cshtml

```
@model List<Client>
<h2>Popis klijenata</h2>
...
@section Scripts{
    <script type="text/javascript">
        alert('Poziv iz view-a Client/Index.cshtml');
    </script>
}
```

Važna napomena: @section ne funkcionra u mehanizmu PartialView – zato se dodaje samo u „pravim“ view-ovima.

jQuery

jQuery je biblioteka koja pruža niz funkcionalnosti za manipulaciju HTML elementima, te je i osnova za veliku većinu modernih javascript plugina. Bazira se na nekoliko bitnijih koncepata koji se često koriste u ASP.NET MVC aplikacijama:

- **jQuery selectors** – mehanizam za određivanje na koje sve elemente se primjenjuje željena manipulacija, jedan
- **jQuery ajax** – pruža mehanizme izvođenja jednostavnih ajax poziva
- **jQuery events** – mogućnost pridruživanja koda uz događaje nad specificiranim HTML elementima
- **jQuery data** – mehanizam vezivanja podataka uz HTML elemente

Pošto (u pravilu) želimo izvršiti kod tek kad je cijeli HTML učitan, koristi se posebna jQuery sintaksa za izvršavanje koda u trenutku kad je DOM (javascript pozadinski objekt vezan uz HTML stranicu) učitan:

```
<script type="text/javascript">
    $(document).ready(function () {
        //Ovdje ide kod koji se treba
        //  izvršiti u trenutku kad je DOM
        //  učitan
    });
</script>
```

Ili kraće:

```
<script type="text/javascript">
    $(function () {
        //Ovdje ide kod koji se treba
        //  izvršiti u trenutku kad je DOM
        //  učitan
    });
</script>
```

Većinu koda koji se izvršava se radi ili kad je cijeli DOM učitan, ili na akciju korisnika.

jQuery selectors

Primjer 1: Nakon što se DOM učita, postaviti pozadinsku boju tablice s id=**tbl-clients** u tabličnom prikazu klijenata na žuto:

Client/Index.cshtml

```
@model List<Client>

<h2>Pregled klijenata</h2>

<table id="tbl-clients" class="table table-cond
    <thead>
        ...
    </tbody>
</table>

@section scripts{
    <script type="text/javascript">
        $(function () {
            $("#tbl-clients").css("background", "yellow");
        });
    </script>

}
```

Kada označujemo specifični element po njegovom id svojstvu koristimo #.

Sve skripte koje želimo da se izvode ćemo staviti u **@section scripts**. Imati na umu da @section scripts nije dostupan u PartialView-ovima.

Primjer 2: Klikom na pojedini redak (<tr>) želimo da se tom retku svi <td> elementi popune sa tekstom '-':

Client/Index.cshtml

```
<table id="tbl-clients" class="table table-condensed">
    <thead>
        ...
    </thead>
    <tbody>
        @foreach (var client in Model)
        {
            <tr onclick="changeText(this)">
                ...
            </tr>
        }

    </tbody>
</table>

@section scripts{
    <script type="text/javascript">
        ...

        function changeText(tr) {
            $(tr).find("td").text("-");
        }
    </script>
}
```

Primjer 3: Prilikom prelaska mišem preko pojedinog retka, želimo taj redak duplicirati i ubaciti na kraj tablice:

Client/Index.cshtml

```
<table id="tbl-clients" class="table table-condensed">
    ...
    <tbody>
        @foreach (var client in Model)
        {
            <tr onclick="changeText(this)" onmouseover="duplicateRow(this)">
                ...
            </tr>
        }

    </tbody>
</table>

@section scripts{
    <script type="text/javascript">
        ...
        function duplicateRow(tr) {
            var dupl = $(tr).clone();
            $("table.table tbody").append(dupl);
        }
    </script>
}
```

Zadatak 9.1

Modificirati Index.cshtml (gdje je tablica s podacima) na način da se dodaju jednostavni javascript efekti uz pomoć dostupnih internet resursa[1]:

1. U trenutku učitavanja DOM-a potrebno je postaviti svojstvo „opacity“ svakog retka na vrijednost „0.5“.
2. U trenutku prelaska mišem preko nekog <tr> elementa potrebno je postaviti mu opacity na 1.0. Kad se miš makne s tog retka, vratiti opacity na „0.5“.
3. Klikom na pojedinu ćeliju (td element) potrebno je postaviti tekst te ćelije i cijelog retka na bold. Ponovnim klikom na istu ćeliju redak treba prestati biti podebljan (bold).

Napomena za zadatak: može se riješiti na više načina. Jedan od načina je ručno na pojedini event postaviti sa jQueryem style vrijednosti; drugi način je definirati <style> tag unutar @scripts dijela i tamo definirati CSS klase koje definiraju opacity/bold, te pomoću jQuery postaviti odgovarajuću klasu u odgovarajućem trenutku.

Zadatak 9.2

Na formi za kreiranje klijenta, kad se fokusira „email“ polje, prikazati dodatni „hint“ za popunjavanje ispod samog polja. Kad se fokus makne, maknuti i hint.

- https://api.jquery.com/focus/

[1] jQuery selectors: http://api.jquery.com/category/selectors/

Zadatak 9.3

Na formi za kreiranje klijenata, ukoliko je bilo koje polje promijenjeno, i pokuša se navigirati sa stranice, tada treba korisnika upozoriti i prikazati mu poruku da postoje promjene koje nisu spremljene. Hint za pretragu: „JS disable navigate if dirty“.

Zadatak 9.4

Na formi za uređivanje klijenata, omogućiti autocomplete za upisivanje adrese. Maps i Places API key se može nabaviti ovdje: https://www.mapbox.com/

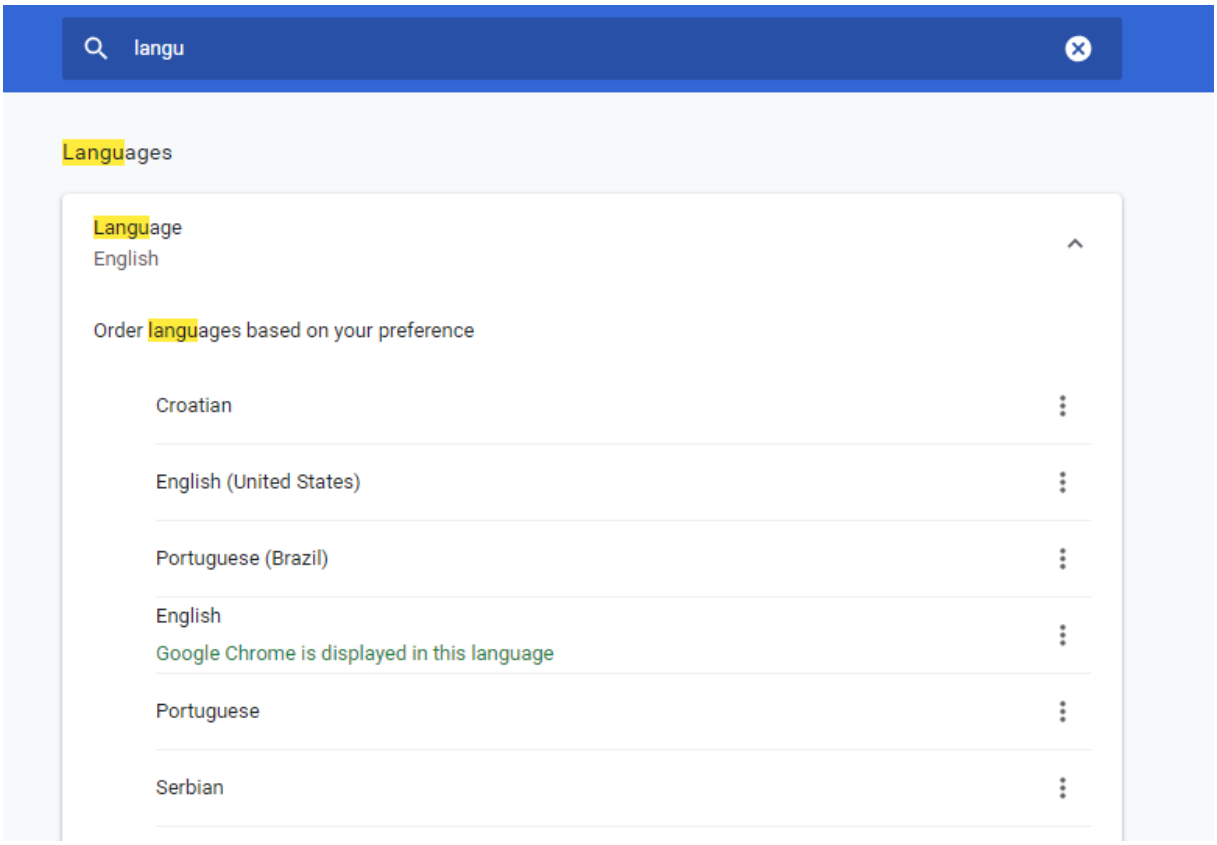
Datumska kontrola – Datepicker

Pri izradi web aplikacija, često se pojavljuje upravo problem s rukovanjem datumskim podacima. Najčešći problem je u formatu datuma, koji je različit ovisno o jeziku kojeg korisnik koristi. U .NET-u, kad govorimo o jeziku, primarno mislimo na CultureInfo objekt, koji se veže uz neku specifičnu državu ili područje. Primjerice, postoji CultureInfo objekt za Englesku (en-GB) ili SAD (en-US), Hrvatsku (hr-HR) i sl. Kad radimo s datumima pravilu je dovoljno koristiti samo dva slova (hr) ili (en) jer takav nacin primjereno definira format datuma.

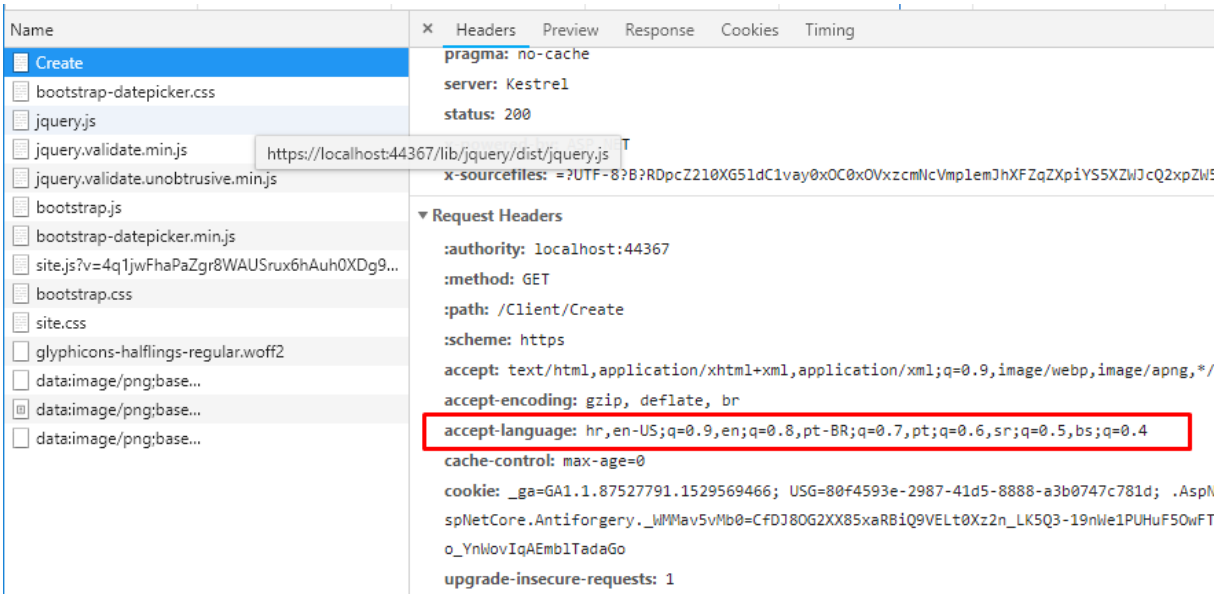
Gdje nastaje problem? Problem nastaje ukoliko korisnik ima u postavkama preglednika postavljeni prioritetni jezik primjerice hr, dok se na serveru očekuje en. Naravno, parsiranje datuma će biti neuspješno, jer US format datuma glasi MM/dd/yyyy dok je na našim područjima uobičajeno dd.MM.yyyy. Dodatni problem se pojavljuje kod definiranja formata datuma na klijentskim komponentama – naime, format datuma definiran u C# jeziku dd.MM.yyyy odgovara klijentskoj inačici dd.mm.yyyy (razlika u malom i velikom slovu za mjesece).

U sklopu ovog kolegija neće se ulaziti u probleme same višejezičnosti i prilagodbe aplikacije za više jezika, ali će se ukazati na potrebne korake u rješavanju problema formata datuma ovisno o klijentskim postavkama.

Klijentski preglednik može imati postavljeno nekoliko jezika koje klijent koristi, često prioritetnim redoslijedom. U Google chrome pregledniku, jezik se definira u „settings“ dijelu:



S obzirom na gornje postavke, prilikom svakog zahtjeva za neku web stranicu, automatski se ta informacija šalje zajedno sa zahtjevom kako bi poslužiteljska aplikacija mogla odlučiti koji jezik da prikaže korisniku:



ASP.NET MVC na .NET 5.0 radnom okviru nudi opciju automatskog prepoznavanja klijentskog jezika upravo po ovom parametru postavljenom u pregledniku:

Startup.cs

```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    ...

    var supportedCultures = new[]
    {
        new CultureInfo("hr"), new CultureInfo("en-US")
    };

    app.UseRequestLocalization(new RequestLocalizationOptions
    {
        DefaultRequestCulture = new RequestCulture("hr"),
        SupportedCultures = supportedCultures,
        SupportedUICultures = supportedCultures
    });

    app.UseEndpoints(endpoints =>
    {
        ...
    });
}
```

Gornjim odsječkom koda je definirano sljedeće:

- Jedini jezici koji su podržani u našoj aplikaciji su Hrvatski i Engleski
- Ako nije drukčije definirano, ili se zahtjeva neki drugi jezik, pretpostavit ćemo hrvatski jezik

VAŽNO: poziv app.UseRequestLocalization() je obvezno napraviti ***PRIJE*** poziva app.UseEndpoints().

Nakon što je u Startup klasi definirana višejezičnost na gornji način, automatski se postavlja ispravni jezik za svaki request i dostupan je bilo gdje u aplikaciji pozivom:

```
System.Globalization.CultureInfo.CurrentCulture
System.Globalization.CultureInfo.CurrentUICulture
```

Culture objekt sadrži niz informacija o tome kakav je format datuma, decimalnih brojeva i slično, a moguće je pregledati detalje kroz debug način rada.

Za datumsku kontrolu moguće je koristiti niz raznih implementacija, a za demonstraciju koristit će se ova implementacija: https://gijgo.com/datepicker/example/bootstrap-4.

Potrebno je samostalno iz dokumentacije osigurati da se kontrola iscrtava i da je format datuma dobar za EN i HR (https://gijgo.com/datepicker/configuration/format).

Zadatak 9.5

Implementirati datumsku kontrolu za novo polje: datum rođenja.

- Dodati polje **DateOfBirth** u klasu **Client**
  - Osigurati da se ispravno definiraju potrebne migracije i baza sinkronizira sa modelom
  - Pošto ima podataka u bazi, polje treba biti *nullable*
- Dodati **app.UseRequestLocalization()** kao što je navedeno u odsječku koda na stranici gore
- Uključiti navedene CSS i JS skripte na ispravnom mjestu u **_Layout** datoteci kako bi bile dostupne u svim view-ovima (može i link koji je dan u dokumentaciji na linku). Takav način uključivanja je CDN link, a nešto više se može pročitati na ovom linku: https://www.cloudflare.com/learning/cdn/what-is-a-cdn/
- Proširiti _CreateOrUpdate partial view sa poljem za datum rođenja po uzoru na dokumentaciju iz biblioteke:
  - Potrebno je u view datoteci na ispravan način definirati format datuma kako bi bio podržan US i HR stil. Proučiti CurrentCulture klasu i iskoristiti neko od dostupnih svojstava. Uzeti u obzir razlike MM i mm.
- Podesiti u pregledniku engleski (US) i zatim hrvatski jezik te osigurati da se može unijeti nova osoba ili urediti datum u ispravnom formatu
  - Osigurati da spremanje podataka uspješno radi neovisno o formatu