ExtJS Tutorijal by Vladica Savić

ExtJS predstavlja JavaScript biblioteku za kreiranje interaktivnih **web 2.0** aplikacija korišćenjem raznih tehnika tipa **AJAX**, **DHTML** i **DOM** skriptinga.

ExtJS uključuje raznolik skup **GUI** (*Grafic User Interface*) kontrola (i tzv. *Widget*-a) za upotrebu unutar web aplikacija kao što su naprimer: Text field, Text area, Date field, List box, Combo Box, Radio i Check box kontrole, HTML editor, Tree kontrole, Tab panele, Toolbare, Flash grafikone, i najčešće upotrebljivanu stvar po meni, Grid kontrolu, i mnoge od ovih kontrola mogu da komuniciraju direktno sa serverside-om koristeći AJAX (*Asynchronous JavaScript and XML*).

No, da ne razglabamo previše sa teorijom, već ćemo videti kako sve to izgleda u praksi kroz jedan mali primer kreiranja telefonskog imenika u kome ću se potruditi da vam što jednostavnije objasnim korak po korak razvojni put jedne proste web 2.0 aplikacije dizajnirane u ExtJS-u, a za serverside ćemo koristiti PHP.

Pre nego što počnemo, postarajte se da preuzmete sa neta, sa zvaničnog ExtJS sajta (www.extjs.com) najnoviju verziju ExtJS biblioteke i možemo početi.

Za ovaj naš primer ja ću kao razvojno okruženje koristiti **Netbeans** (možete bilo koji tekstualni editor, **Notepad++** ili dr.) ja sam se opredelio za netbeans čisto zbog formatiranja koda, a za rad na serverside-a ću koristiti **WAMP server** jer će mi biti potrebna podrška za rad sa **PHP**-om i **MySQL** bazom u koju ćemo da smeštamo podatke vezane za naš imenik (o serverside skriptama neće biti previše reči u ovom tutorijalu, jer se pretpostavlja da već posedujete znanje potrebno za programiranje istih u nekom od programskih jezika tipa PHP, C# ili nekom trećem... koji će nam služiti za obradu podataka).

Primer aplikacije koja je opisana u tutorijalu možete videti na adresi:
http://vladicasavic.iz.rs/extjs/

Komletan source tutorijala možete preuzeti sa lokacije:
http://vladicasavic.iz.rs/extjs/download.php?file=ExtJS-Tutorijal-SourceCode.zip

Prvi korak – Kreiranje baze

Koriscićemo prostu strukturu baze koja će u mom primeru izgledati ovako:

Ime baze: „Imenik“
Ime tabele: „Kontakti“
(bez navodnika)

Struktura tabele će nam izgledati ovako:

Naziv polja	**Tip polja**	**Dodatno**
KontaktID	**Int (10)**	**PK**, **AutoInc**
Telefon	**VarChar(30)**	
Ime	**VarChar(15)**	
Prezime	**VarChar(15)**	
Adresa	**VarChar(100)**	

...naravno vi je možete izmeniti, ali imajte u vidu da ćete sami morati da vodite računa o tome u daljim koracima.

Drugi korak – Kreiranje layouta

Najpre ćemo da kreiramo osnovni layout naše aplikacije i mali loader obzirom da ext-ove biblioteke same po sebi nisu baš preterano ''male'' po veličini (pogotovu ako uzmemo u obzir još uvek velik broj ljudi u našoj okolini koji još uvek koriste spor internet) kako bi imali uvid u to dokle je stiglo učitavanje naše aplikacije.

Kao što smo već rekli, obzirom da je ExtJS ništa drugo do malo naprednije pisan JavaScript na isti način kao i bilo koje druge skripte se uključuje u okviru HTML strane kao što možete videti u primeru ispod:

Index.html

[code]

```
<html>
<head>
       <title>Telefonski imenik - 1.0 Alfa - Programmed in ExtJS by Vladica Savic</title>
<link href="css/site.css" rel="stylesheet" type="text/css" />
</head>
<body>
<div id="loading-mask" style="">
</div>
<div id="loading">
<center>
<div>
<img src="slike/ajax-loader.gif" width="32" height="32" style="margin-right:8px;float:left;vertical-align: top;"/>
       Telefonski imenik - 1.0 - <a href="http://www.vladicasavic.iz.rs">www.vladicasavic.iz.rs</a>
<br />
<span id="loading-msg" style="float: left;">
        Ucitavanje slika i stilova...
</span>
</div>
</center>
</div>
        <!-- include ext-all.css -->
    <link rel="stylesheet" href="js/ext/resources/css/ext-all.css" />
    <!-- include ext-base.js -->
    <script type="text/javascript">
             document.getElementById('loading-msg').innerHTML = 'Ucitavanje API-a';
    </script>
    <script type="text/javascript" src="js/ext/adapter/ext/ext-base.js"></script>
    <!-- include ext-all.js -->
    <script type="text/javascript">
             document.getElementById('loading-msg').innerHTML = 'Ucitavanje ExtJS-a';
    </script>
    <script type="text/javascript" src="js/ext/ext-all.js"></script>

    <script type="text/javascript">
             document.getElementById('loading-msg').innerHTML = 'Ucitavanje panela...';
    </script>

    <script type="text/javascript" src="js/toolbar/GlavniToolbar.js"></script>
    <script type="text/javascript" src="js/gridovi/Grid.js"></script>
    <script type="text/javascript" src="js/prozori/KontaktProzor.js"></script>
    <script type="text/javascript" src="js/Aplikacija.js"></script>

    <script type="text/javascript">
             document.getElementById('loading-msg').innerHTML = 'Aplikacija se pokrece...';
    </script>
</body>
</html>
```

[/code]

(Izvinjavam se zbog smanjene preglednosti koda, ali moralo je ovako…)

Kao što ste i sami primetili, samo smo učitali skripte koje su nam potrebne za dalji rad sa ExtJS-om u svoju HTML stranu, i kako smo koju učitavali vršili smo ispis na ekranu kako bi imali indicator stanja dokle je stiglo učitavanje naše aplikacije.

Sve skripte koje smo učitali su ExtJS-ove osim skripte pod imenom Aplikacija.js koja je skripta gde ćemo da kreiramo aplikaciju i skripti GlavniToolbar.js, Grid.js, kao i KontaktProzor.js unutar kojih ćemo da generišemo objekte koje ćemo koristiti u Aplikacija.js fajlu.

Napomena!
Obzirom da je ovo sam početak i da nemamo ništa u svojoj aplikacici, i obzirom da ćemo kasnije kreirati jos par java skripti (ja cu ih uključiti na kraju ovog tutorijala u HTML strani, ali je važno da to napomenem na samom početku) čije ćemo objekte da koristimo moraćemo i te skripte da uključimo u našu HTML stranu i to vodeći računa o njihovoj hijerarhiji, tj. redosledu prema kojem ćemo kreirati objekte i prema kome ćemo ih koristiti dalje u aplikaciji.

Glavni layout

Naša aplikacija će se prikazivati u vidu prozora koji će u sebi sadrzati toolbar sa menijima za radnje tipa dodavanje novog kontakta, brisanje već unetih kontakata, i pozivanje dijalog prozora za izmenu odredjenih kontakata, i centralnog dela u kome će se nalaziti grid komponenta za prikaz unetih kontakata.

Za početak ćemo napraviti samo okvir prozora kome ćemo kasnije da dodamo toolbar sa gore navedenim menijima i ekranskim tasterima, i u koji ćemo da ubacimo grid komponentu.

To ćemo raditi u fajlu pod nazivom "*Aplikacija.js*" i on će izgledati ovako:

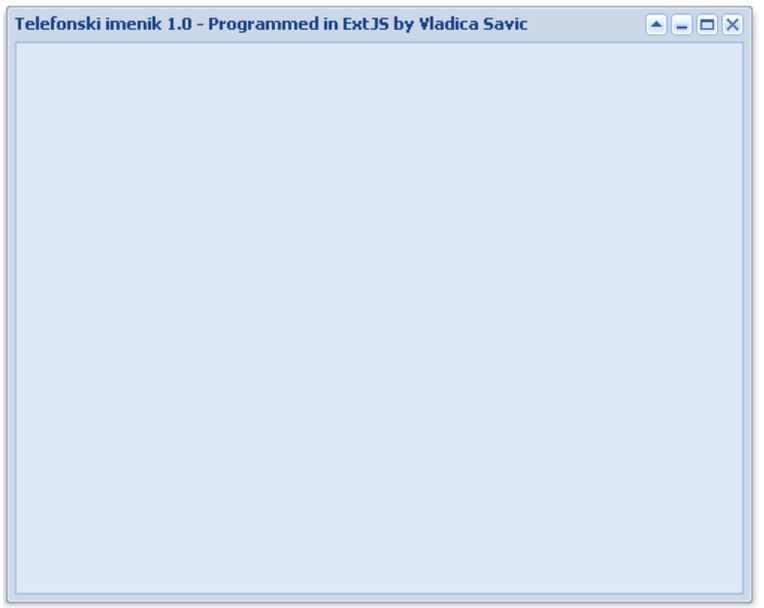
Aplikacija.js

[code]

```
Ext.namespace('TelefonskiImenik');

Ext.BLANK_IMAGE_URL = 'scripts/ext/resources/images/default/s.gif';
Ext.onReady(function() {
   try {
                document.execCommand('BackgroundImageCache', false, true);
          } catch (ex) { }
   Ext.QuickTips.init();

   TelefonskiImenik.Application = function() {
      var hideMask = function () {
         Ext.get('loading').remove();

                kreirajGlavniProzor();
      }
      hideMask.defer(1000);

      function kreirajGlavniProzor() {
         var glavniProzor = new Ext.Window({
                        title: 'Telefonski imenik 1.0 - Programmed in ExtJS by Vladica Savic',
                        width: 500,
                        height: 400,
                        collapsible: true,
                        minimizable: true,
                        maximizable: true,
            items: [ ]
         });
         glavniProzor.show();
      };
   } ();
});
```

[/code]

I kao rezultat imaćemo prozor kao na slici ispod.



To će nam biti osnovni layout u kome ćemo da smeštamo ostale komponente (toolbar, grid).

Naša aplikacija će sadržati i toolbar u kome ćemo imati par dugmadi za dodavanje novih, brisanje i editovanje postojećih kontakata kao i jedno dugme za izlaz iz aplikacije. Pošto ćemo se u ovom malom tutorijalu truditi da programiramo na „pravilan“ način kao da je reč o nekoj velikoj aplikaciji koristićemo tzv. konstruktore, kako bi naš kod bio podeljen u logičke celine lakše i razumljivije drugima koji bi se kasnije recimo priključili u projekat i nastavili njegov dalji razvoj ili eventualne izmene.

GlavniToolbar.js

[CODE]

```
Ext.namespace('TelefonskiImenik');

TelefonskiImenik.GlavniToolbar = function() {
    var self = this;

    var akcija  = {
        DodajKontakt: new Ext.Action({
            text: 'Dodaj',
            handler: DodajKontakt,
            iconCls: 'akcija-dodaj',
            scale: 'medium'
        }),
        ObrisiKontakt: new Ext.Action({
            text: 'Obrisi',
            handler: ObrisiKontakt,
            iconCls: 'akcija-obrisi',
            scale: 'medium'
        }),
        IzmeniKontakt: new Ext.Action({
            text: 'Izmeni',
            handler: IzmeniKontakt,
            iconCls: 'akcija-izmeni',
            scale: 'medium'
        }),
        ZatvoriGlavniProzor: new Ext.Action({
            text: 'Izlaz',
            handler: ZatvoriGlavniProzor,
            iconCls: 'akcija-izlaz',
            scale: 'medium'
        })
    };

    function ZatvoriGlavniProzor() {
        self.fireEvent('ZatvoriGlavniProzor');
    }

    function DodajKontakt()
    {
            self.fireEvent('DodajKontakt');
      }
    function ObrisiKontakt()
    {
            self.fireEvent('ObrisiKontakt');
      }
    function IzmeniKontakt()
    {
            self.fireEvent('IzmeniKontakt');
      }

    TelefonskiImenik.GlavniToolbar.superclass.constructor.call(this, {
        items: [
        akcija.DodajKontakt,
        '-',
```

```
        akcija.ObrisiKontakt,
        '-',
        akcija.IzmeniKontakt,
        '->',
        akcija.ZatvoriGlavniProzor
        ]
    });
};

Ext.extend(TelefonskiImenik.GlavniToolbar, Ext.Toolbar, {
    initComponent: function() {
        TelefonskiImenik.GlavniToolbar.superclass.initComponent.apply(this, arguments);
        this.addEvents('DodajKontakt');
        this.addEvents('ObrisiKontakt');
        this.addEvents('IzmeniKontakt');
        this.addEvents('ZatvoriGlavniProzor');
    }
});
```

[/CODE]

Šta primećujemo u ovom delu koda?

Primećujemo da imamo par definisanih akcija, opisanih tekstom, ikonom, tj. stilom, i tzv. handlerom, tj. funkcijom koja ce biti pokrenuta pozivanjem neke od ovih akcija. Takodje, svaka od ovih akcija predstavlja po jedan taster koji je predstavljen kao element konstruktora. Na interface-u možemo odvojiti tastere takozvanim separatorima koji se predstavljaju sa `'-'`. Takodje, možemo odvojiti grupu elemenata postavljajući je na drugi kraj toolbara separatorom `'->'` koji kaže da se sledeći element toolbar-a smesti skroz uz desnu stranu istog toolbara.

Takodje, pošto svaki od taster koristi odgovarajuću css klasu kako bi učitali željenu ikonicu moramo napraviti i css fajl koji ćemo uključiti u naš projekat, i napraviti odgovarajuće css klase, za naš slučaj to su sledeće klase: `'akcija-dodaj'`, `'akcija-obrisi'`, `'akcija-izmeni'`, `'akcija-izlaz'`. Što zači da bi css fajl trebao da sadrzi sledeći kod vezan za tastere u toolbar-u:

[CODE]

```
/* Slike tastera glavnog toolbara */
.akcija-dodaj
{
    background-image: url(../slike/akcija-dodaj.png) !important;
}
.akcija-obrisi
{
    background-image: url(../slike/akcija-obrisi.png) !important;
}
.akcija-izmeni
{
    background-image: url(../slike/akcija-izmeni.png) !important;
}
.akcija-izlaz
{
    background-image: url(../slike/akcija-izlaz.png) !important;
}
```

[/CODE]

Napomena!
Vodite računa o putanjama i nazivima ikona koje želite da dodate kako se ne bi desilo da na interface-u dobijete taster bez ikone.

Nakon dodavanja ***GlavniToolbar.js*** skripte u listu skripti na nasoj index strani (ukljucenu pre ***Aplikacija.js*** skripte) kreiranja toolbar objekta unutar Aplikacija.js fajla radi se na sledeći način:

[CODE]

```
var glavniToolbar = new TelefonskiImenik.GlavniToolbar();
```

[/CODE]

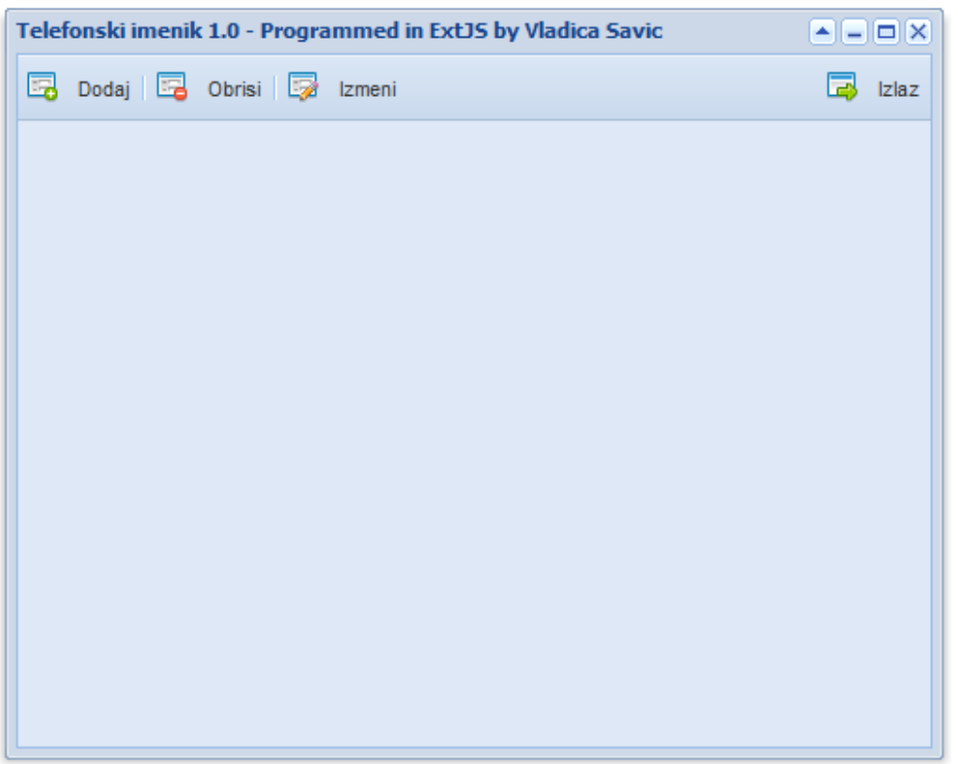
I postavljanja tbar (toolbar) property-a objektu "glavniProzor" na sledeći način:

[CODE]

```
var glavniProzor = new Ext.Window({
                title: 'Telefonski imenik 1.0 - Programmed in ExtJS by Vladica Savic',
                width: 500,
                height: 400,
                collapsible: true,
                minimizable: true,
                maximizable: true,
                maximized: false,
                items: [ ],
                tbar: glavniToolbar
            });
```

[/CODE]

Pokretanjem aplikacije dobijamo prizor kao na slici ispod.



Naša aplikacija će takodje sadržati još jedan prozor koji će nam služiti za unos novih kontakata i izmenu postojećih. Njega ćemo definisati u fajlu pod imenom KontaktProzor.js koji ćemo takodje uključiti u Index.html stranu, opet vodeći računa o tome da se on učita pre Aplikacija.js skripte.

KontaktProzor.js

[CODE]

```
Ext.namespace('TelefonskiImenik');


TelefonskiImenik.KontaktProzor = function(rezim) {
    var self = this;
    var naslovProzora = (rezim=='izmena')?"Izmeni informacije":"Unos novog kontakta";
    var akcija  = {
        SacuvajKontaktInformacije: new Ext.Action({
            text: 'Sacuvaj',
            handler: SacuvajKontaktInformacije
        }),
        ZatvoriKontaktProzor: new Ext.Action({
            text: 'Zatvori',
            handler: ZatvoriKontaktProzor
        })
    };

    function ZatvoriKontaktProzor(akcija, e)
    {
        self.close();
    }
    function SacuvajKontaktInformacije(akcija, e)
    {
        self.fireEvent('SacuvajKontaktInformacije');
    }

    TelefonskiImenik.KontaktProzor.superclass.constructor.call(this, {
        title: naslovProzora,
        width: 290,
        height: 290,
        resizable: false,
        split: true,
        modal: true,
        items: [{
                    xtype: 'label',
                    style: 'margin: 5;',
                    text: 'Ime:'
                },{
                    xtype:'textfield',
                    width: 265,
                    name: 'Ime',
                    id: 'Ime',
                    style: 'margin: 5px;'
                },{
                    xtype: 'label',
                    style: 'margin: 5;',
                    text: 'Prezime:'
                },{
                    xtype:'textfield',
                    width: 265,
                    name: 'Prezime',
                    id: 'Prezime',
                    style: 'margin: 5px;'
                },{
```

```
                    xtype: 'label',
                    style: 'margin: 5;',
                    text: 'Adresa:'
                },{
                    xtype:'textarea',
                    name: 'Adresa',
                    id: 'Adresa',
                    width: 265,
                    style: 'margin: 5px',
                    grow: false,
                    preventScrollbars:true
                },{
                    xtype: 'label',
                    style: 'margin: 5;',
                    text: 'Telefon:'
                },{
                    xtype:'textfield',
                    width: 265,
                    name: 'Telefon',
                    id: 'Telefon',
                    style: 'margin: 5px;'
                }
        ],
        buttons: [
            akcija.SacuvajKontaktInformacije,
            akcija.ZatvoriKontaktProzor
        ]

    });
};

Ext.extend(TelefonskiImenik.KontaktProzor, Ext.Window, {
    initComponent: function() {
        TelefonskiImenik.KontaktProzor.superclass.initComponent.apply(this, arguments);
        this.addEvents('SacuvajKontaktInformacije');
    }
});
```
[/CODE]

Šta možemo da primetimo ovde... Obzirom da ćemo ovaj isti objekat-prozor da koristimo i za unos novih i za izmenu postojećih kontakata kao parameter f-ji koja služi za kreiranje ovog prozora slaćemo i parameter koji služi kao identifikatr “režima” prema kome ćemo da prikažemo korisniku na interface-u prozor prilagodjen novom unosu, ili izmeni kontakata.

Takodje, možemo da primetimo da ovaj objekat ima tzv item-e, tj. da sadrži druge objekte, u ovom konkretnom slučaju sadrži par labela koje nam služe kao identifikatori korisniku koje je polje čemu namenjeno, text field-ova i text area u kojima će korisnik da unosi podatke koji će se upisivati u bazu.

Postoji dva načina za kreiranje objekata / item-a koje neki drugi objekat može da sadrži. Ja sam item-e onde definisao unutar samog objekta na sledeći način:

[CODE]

```
…
{

  xtype:'textarea',
  name: 'Adresa',
  id: 'Adresa',
  width: 265,
  style: 'margin: 5px',
  grow: false,
  preventScrollbars:true
}
…
```

[/CODE]

... na ovaj način mi dakle definišemo preko xtype property-a o kom tipu objekta se radi, u ovom slucaju radi se o text area objektu, definišemo mu ime, id, stil, i druga adekvatna svojstva tom tipu objekta.

Drugi način je da kreiramo zaseban objekat koji samo dodamo nekom drugom element po potrebi.
Npr. ekvivalent gore definisanom objektu može se napraviti i ovako:

[CODE]

```
…
var nekiTextAreaObjekat = Ext.form.TextArea({
      name: 'Adresa',
      id: 'Adresa',
      width: 265,
      style: 'margin: 5px',
      grow: false,
      preventScrollbars:true
});
…
```

[/CODE]

...i kasnije samo dodamo u listu item-a promenjivu ***nekiTextAreaObjekat*** i na taj način možda dobijemo i donekle preglednije rešenje.

Pored prethodno kreiranih objekata, jedna od najkorisnijih po meni i najčešće upotrebljivanih komponenti je tzv. grid komponenta. Ona će u našem slučaju služiti za prikaz svih prethodno unetih podataka. Nju ćemo definisati u *Grid.js* fajlu na sledeći način:

Grid.js

[CODE]

```
Ext.namespace('TelefonskiImenik');

TelefonskiImenik.KontaktGrid = function() {

    var paginacija = 20;

    var storeRecord = new Ext.data.Record.create([
        { name: 'KontaktId', type: 'int' },
        { name: 'Ime' },
        { name: 'Prezime' },
        { name: 'Adresa' },
        { name: 'Telefon' }
    ]);

    var dataStore = new Ext.data.Store({
        url: 'ServerSide/Obradi.php',
        baseParams: {
            action: 'ucitajKontakte',
            limit: paginacija
        },
        reader: new Ext.data.JsonReader(
            {
                root: "podaci",
                totalProperty: "ukupnoKontakata"
            },
            storeRecord
        ),
        listeners: {
            'loadexception': {
                fn: function(data) {
                    var responseText = data;
                    if(responseText == '') {
                         Ext.MessageBox.alert('Informacija', 'Nema unetih kontakata u bazi.');
                    }
                }
            }
        }
    });

    TelefonskiImenik.KontaktGrid.superclass.constructor.call(this, {
        title: '',
        store: dataStore,
        loadMask: false,
        border: false,
        frame: false,
        stripeRows: true,
        bbar: new Ext.PagingToolbar({
            pageSize: paginacija,
            store: dataStore,
            displayInfo: true,
            displayMsg: 'Prikazano {0} - {1} od {2} kontakta',
            emptyMsg: "0 - 0 od 0"
        }),
        sm: new Ext.grid.RowSelectionModel({
            singleSelect: true
        }),
```

```
        columns: [
            {
                id: 'Ime',
                width: 100,
                header: 'Ime',
                dataIndex: 'Ime',
                sortable: true
            },
            {
                header: 'Prezime',
                width: 100,
                dataIndex: 'Prezime',
                sortable: true
            },
            {
                header: 'Adresa',
                width: 260,
                dataIndex: 'Adresa',
                sortable: true
            },
            {
                header: 'Telefon',
                width: 100,
                dataIndex: 'Telefon',
                sortable: true
            }
        ]
    });
};

Ext.extend(TelefonskiImenik.KontaktGrid, Ext.grid.GridPanel, {
    initComponent: function() {
        TelefonskiImenik.KontaktGrid.superclass.initComponent.apply(this, arguments);
    },
    onRender: function() {
        TelefonskiImenik.KontaktGrid.superclass.onRender.apply(this, arguments);
    }
});
```
[/CODE]

Šta je ono karakteristično što primećujemo u ovom delu koda... Ovde primećujemo da imamo jedan dataStore objekat koji nam u suštini predstavlja objekat u koji smeštamo podatke koje ćemo prikazivati u našem gridu. Definiciju objekata koje sadrži dataStore predstavlja dataRecord. Takodje, pošto ćemo da sa servera dovlačimo podatke u json formatu data store će koristiti Json Reader za čitanje tih istih podataka. Url parameter data store objekta predstavlja koji handler će se pozvati za dobijanje podataka sa server, a `baseParams` predstavlja deo u kome definišemo koje ćemo dodatne podatke slati serverskoj skripti. U našem slučaju pošto nam se ceo pozadinski kod nalazi u istoj skripti a mi želimo da izvršimo samo onaj deo koji će nam vratiti podatke iz baze prosledjujemo parametar *actions* koji dalje na serverskoj strani hvatamo kako bi skripta izvršila samo taj potreban deo koda i *limit* koji nam predstavlja koliko rezultata želimo da nam upit vrati (kako bi izbegli preopterećivanje servera i bespotrebno vraćanje svih rezultata sql upita) koji se odnosi na prethodno unete kontakte.

Takodje, specifičan deo našeg grida je *RowSelectionModel* koji je postavljen na singleSelect što znači da samo jedan red može biti selektovan, mada se to može izmeniti po potrebi da u isto vreme korisnik može da selektuje više redova, ali u našem slučaju to nije potrebno.

Takodje, karakteristično za grid je bottom bar (bbar) mada se može koristiti i u toolbar-u *PagingToolbar* koji služi za paginaciju podataka u gridu.

Takodje još jedan karakterističan deo za grid je deo za definisanje kolona koja se prikazuju u njemu. Pored standardnog izgleda koji je inače korišćen za naš slucaj, moguće je pridružiti tzv. renderer koji omogućuje da se odredjena polja po potrebi drugačije prikazuju na interfejsu, recimo da se u ćeliji grid-a prikaže combobox, checkbox, datefield, color picker, ili neki custom html.

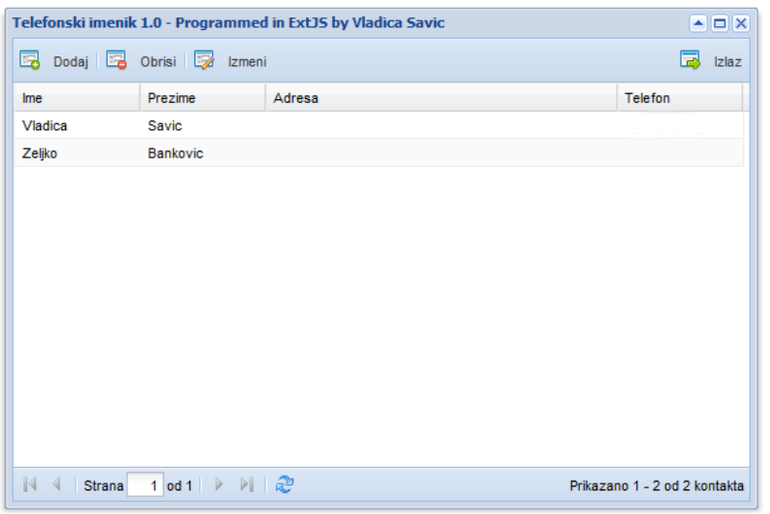
Neki svoj custom template ovjekat (ovo je generalno, ne vezano za grid) možete definisati na sledeći način:

[CODE]

```
var vasTemplateObjekat = new Ext.XTemplate(
          '<div>Neki vaš HTML</div>',
          '<p>...koji se može podesiti kako god želite!</p>'
);
```

[/CODE]

Nakon dodatih skripti, i podešavanja na gore prikazan način dobili bi interface kao na slici ispod:



Nakon što smo prethodno ukratko opisali objekte koje ćemo koristiti, u fajlu *Aplikacija.js* napravićemo adekvatne objekte i postaviti ih kao odgovarajuće item-e u naš glavni layout. Evo kako izgleda konačna verzija datog fajla, obzirom da je "pozamašan" ja ću prokomentarisati samo neke delove ovog fajla, koji su vama bitni, tj. ono što je na neki način korisno, i specifično za sam ext, o logici aplikacije ću samo ukratko, jer smatram da svako ko se našao ovde da već ima neko ranije iskustvo u programiranju u nekom od drugih programskih jezika i da objašnjenja na tu temu nisu neophodna.

Aplikacija.js

[CODE]

```
Ext.namespace('TelefonskiImenik');
Ext.BLANK_IMAGE_URL = 'scripts/ext/resources/images/default/s.gif';
Ext.onReady(function() {
    try {
        document.execCommand('BackgroundImageCache', false, true);
    } catch (ex) { }
    Ext.QuickTips.init();

    TelefonskiImenik.Application = function() {
        var hideMask = function () {
            Ext.get('loading').remove();

            kreirajGlavniProzor();
        }
        hideMask.defer(1000);

       //Overide standardnog teksta dugmadi i sl. extjs kontrola
       //kako bi bili lokalizovani za naš jezik...
       //-------------------------------------------------------
        Ext.MessageBox.buttonText = {
            ok      : "OK",
            cancel  : "Otkazi",
            yes     : "Da",
            no      : "Ne"
        };

        Ext.apply(Ext.PagingToolbar.prototype, {
            beforePageText  : "Strana",
            afterPageText   : "od {0}",
            firstText       : "Prva strana",
            prevText        : "Prethodna strana",
            nextText        : "Sledeca strana",
            lastText        : "Poslednja strana",
            refreshText     : "Osvezi"
        });

        Ext.apply(Ext.grid.GridView.prototype, {
            sortAscText     : "Sortiraj rastuce",
            sortDescText    : "Sortiraj opadajuce",
            columnsText     : "Kolone"
        });
        //-------------------------------------------------------
        function kreirajGlavniProzor() {
            var glavniToolbar   = new TelefonskiImenik.GlavniToolbar();
            var kontaktGrid     = new TelefonskiImenik.KontaktGrid();

            kontaktGrid.getStore().load({
                params:{
                    start: 0,
                    limit: 20
                }
            });
```

```
glavniToolbar.on(
    'ZatvoriGlavniProzor',function(e, target) {
        ZatvoriGlavniProzor();
    });
glavniToolbar.on('DodajKontakt',function(e, target) {
    DodajKontakt();
});
glavniToolbar.on('ObrisiKontakt',function(e, target) {
    ObrisiKontakt();
});
glavniToolbar.on('IzmeniKontakt',function(e, target) {
    IzmeniKontakt();
});

var glavniProzor = new Ext.Window({
    title       : 'Telefonski imenik 1.0 - Programmed in ExtJS by Vladica Savic',
    width       : 600,
    height      : 400,
    layout      :'fit',
    collapsible : true,
    maximizable : true,
    maximized   : false,
    items: [
        kontaktGrid
    ],
    tbar        : glavniToolbar
});
glavniProzor.show();

function ZatvoriGlavniProzorAkcija(buttonId, text, opt)
{
    if(buttonId=="yes")
    {
        glavniProzor.close();
    }
}

function ZatvoriGlavniProzor(){
    Ext.Msg.show({
        title   : 'Izlaz',
        msg     : 'Da li stvarno zelite da napustite aplikaciju?',
        buttons : Ext.Msg.YESNO,
        fn      : ZatvoriGlavniProzorAkcija,
        width   : 300
    });
}

function SacuvajKontaktInformacije(rezim){

    var ime     = Ext.get('Ime').dom.value;
    var prezime = Ext.get('Prezime').dom.value;
    var adresa  = Ext.get('Adresa').dom.value;
    var telefon = Ext.get('Telefon').dom.value;

    if(rezim=='izmena'){
        //Deo koda vezan za izmenu podataka o trenutno selektovanom kontaktu
        IzmeniKontaktAkcija(ime, prezime, adresa, telefon);
    }else{
        //Deo koda za insert novog kontakta u imenik
        DodajKontaktAkcija(ime, prezime, adresa, telefon);
    }
}
```

```
        function ObrisiKontakt(){
            //Proveravamo da li je selektovan neki od kontakata da bi mogli da ga obrisemo
            if (kontaktGrid.getSelectionModel().hasSelection())
            {
                Ext.Msg.show({
                    title   : 'Brisanje',
                    msg     : 'Da li stvarno zelite da obrisete kontakt iz imenika?',
                    buttons : Ext.Msg.YESNO,
                    fn      : ObrisiKontaktAkcija,
                    width   : 320
                });
            }else{
                Ext.Msg.show({
                    title   :'Greska',
                    msg     :'Niste izabrali kontakt koji zelite da obrisete!',
                    buttons : Ext.Msg.OK
                });
            }
        }

        function DodajKontakt(){
            var kontaktProzor = new TelefonskiImenik.KontaktProzor("dodavanje");
            kontaktProzor.show();
            kontaktProzor.on('SacuvajKontaktInformacije',function(e, target) {
                SacuvajKontaktInformacije("dodavanje");
                kontaktProzor.close();
            });
        }

        function IzmeniKontakt(){
            if (kontaktGrid.getSelectionModel().hasSelection())
            {
                var kontaktProzor = new TelefonskiImenik.KontaktProzor("izmena");
                kontaktProzor.show();

                //...nakon otvaranja kontakt prozora automatski popunjavamo polja
                //vrednostima iz selektovanog reda iz grida

                Ext.get('Ime').dom.value        =
kontaktGrid.getSelectionModel().getSelections()[0].data.Ime;
                Ext.get('Prezime').dom.value    =
kontaktGrid.getSelectionModel().getSelections()[0].data.Prezime;
                Ext.get('Adresa').dom.value     =
kontaktGrid.getSelectionModel().getSelections()[0].data.Adresa;
                Ext.get('Telefon').dom.value    =
kontaktGrid.getSelectionModel().getSelections()[0].data.Telefon;

                //...pristupanje i postavljanje vrednosti nekom objektu
                //Ext.get('idElementa').dom.value = 'neka vrednost';
                //...pristupanje i postavljanje vrednosti nekom objektu na drugaciji nacin
                //Ext.get('idElementa').set({
                //    value: 'neka vrednost'
                //});

                kontaktProzor.on('SacuvajKontaktInformacije',function(e, target) {
                    SacuvajKontaktInformacije("izmena");
                    kontaktProzor.close();
                });
            }else{
                Ext.Msg.show({
                    title   :'Greska',
                    msg     :'Niste izabrali kontakt koji zelite da izmenite!',
                    buttons : Ext.Msg.OK
```

```
        });
    }
}

function IzmeniKontaktAkcija(imeKontakta, prezimeKontakta, adresaKontakta, telefonKontakta)
{
    kontaktGrid.getGridEl().mask('Snimanje izmena u toku...');
    var idKontakta = kontaktGrid.getSelectionModel().getSelections()[0].data.KontaktId;
    Ext.Ajax.request({
        url: 'ServerSide/Obradi.php',
        method: 'POST',
        params: {
            action      : 'izmeniKontakt',
            kontaktID   : idKontakta,
            ime         : imeKontakta,
            prezime     : prezimeKontakta,
            adresa      : adresaKontakta,
            telefon     : telefonKontakta
        },
        success: function(response, options) {
            kontaktGrid.getStore().load({
                params:{
                    start: 0,
                    limit: 20
                }
            });
            kontaktGrid.getGridEl().unmask(true);
        }
    });
}

function DodajKontaktAkcija(imeKontakta, prezimeKontakta, adresaKontakta, telefonKontakta){
    kontaktGrid.getGridEl().mask('Unose se novi podaci...');
    Ext.Ajax.request({
        url: 'ServerSide/Obradi.php',
        method: 'POST',
        params: {
            action  : 'dodajKontakt',
            ime     : imeKontakta,
            prezime : prezimeKontakta,
            adresa  : adresaKontakta,
            telefon : telefonKontakta
        },
        success: function(response, options) {
            kontaktGrid.getStore().load({
                params:{
                    start: 0,
                    limit: 20
                }
            });
            kontaktGrid.getGridEl().unmask(true);
        }
    });
}

function ObrisiKontaktAkcija(buttonId, text, opt){
    if(buttonId=="yes")
    {
        kontaktGrid.getGridEl().mask('Podaci se osvezavaju ...');
        Ext.Ajax.request({
            url: 'ServerSide/Obradi.php',
            method: 'POST',
            params: {
```

```
                            action      : 'obrisiKontakt',
                            kontaktID   :
kontaktGrid.getSelectionModel().getSelections()[0].data.KontaktId
                        },
                        success: function(response, options) {
                            kontaktGrid.getStore().load({
                                params:{
                                    start: 0,
                                    limit: 20
                                }
                            });
                            kontaktGrid.getGridEl().unmask(true);
                        }
                    });
                }
            }
        }
    } ();
});
```
[/CODE]

Šta je zanimljivo u ovoj skripti?
Po meni, u ovoj skripti za ovaj mali tutorijal, najzanimljivija je ***Ext.Ajax.request*** metoda.

[CODE]
```
Ext.Ajax.request({
                    url: 'ServerSideSkripta.php',
                    method: 'POST',
                    params: {
                        //Dodatni parametri koji se salju server side skripti koja je navedena u url-u
                    },
                    success: function(response, options) {
                        //deo namenjen za izvrsenje dela koda nakon uspesne ajax obrade
                        //...u ovom delu koda mozete takodje videti response sa servera
                        //a koji opet i dalje mozete obradjivati po zelji.
                    }
                });
```
[/CODE]

Takodje, tu je i deo za interakciju s korisnikom. U zavisnosti od neke akcije ponekad je potrebno obavestiti korisnika da je željena akcija obavljena ili nije, ili da se korisnik upita da li je siguran da zaista želi da izvrši akciju koju je izabrao, npr. akcija brisanja nekog kontakta u našem slučaju. Ext nudi zanimljive dijaloge.

Evo primera kako se koriste dijalozi.

[CODE]
```
Ext.Msg.show({
                    title   : 'Neki vas title poruke',
                    msg     : 'Neki tekst koji zelite da prikazete korisniku',
                    buttons : Ext.Msg.YESNO, //Moguci tasteri: Ext.Msg.OK, Ext.Msg.YESNOCANCEL ...
                    //Funkcija koja se pokrece na klik na taster u okviru dijaloga
                    fn      : VasaFunkcija,
                    width   : 320
            });
```
[/CODE]
...detekcija tastera koji je bio kliknut od strane korisnika u okviru dijaloga koji je prikazan ide po parametru `buttonId` u funkciji koja je navedena za dijalog (...dakle ***`VasaFunkcija(buttonId,...)`***).

Pozadinski kod – PHP

Ceo pozadinski kod koji ćemo koristiti za upis, čitanje podataka iz baze, izmenu i brisanje (tzv. **CRUD**) stavićemo u skriptu pod imenom „Obradi.*php*“ koji će izgledati ovako:

Obradi.php

[CODE]

```
<?php

/*
 * Kod za citanje podataka/kontakata iz baze, izmenu, brisanje, i unos
 * ...deo koda manje vazan za ceo ovaj tutorijal.
 */
if(isset($_POST["action"])) {
    //Promeniti parametre za pristup bazi u odgovarajuce po potrebi
    $konekcija = mysql_connect("localhost", "root", "") or die (mysql_error ());
    $baza = mysql_select_db ("imenik") or die (mysql_error ());

    $akcija = $_POST["action"];
    switch ($akcija) {
        case "ucitajKontakte": {
                $sql = mysql_query ("SELECT * FROM kontakti") or die (mysql_error());
                $ukupno = mysql_num_rows($sql);
                $sqlRezultat = mysql_query (
                        "SELECT * FROM kontakti ORDER BY KontaktId LIMIT ".$_POST['start'].",
".$_POST['limit']."")
                        or die (mysql_error ());
                $podaci = array();
                while ($kontakt=mysql_fetch_object($sqlRezultat)) {
                    $podaci [] = $kontakt;
                }
                echo '({"ukupnoKontakata":"'.$ukupno.'","podaci":'.json_encode($podaci).'})';
            }
            break;
        case "dodajKontakt": {
                $ime        = $_POST["ime"];
                $prezime    = $_POST["prezime"];
                $adresa     = $_POST["adresa"];
                $telefon    = $_POST["telefon"];

                $sqlDodajKontakt = 'INSERT INTO kontakti (
                                        Ime,
                                        Prezime,
                                        Adresa,
                                        Telefon)
                                    VALUES (
                                        "'.$ime.'",
                                        "'.$prezime.'",
                                        "'.$adresa.'",
                                        "'.$telefon.'")';
                mysql_query("SET NAMES utf8");
                echo $sqlDodajKontakt;
                if(!mysql_query($sqlDodajKontakt)) {
                    die('Greska! Kontakt nije unet. Detalji greske: '.mysql_error());
                }
            }
            break;
        case "izmeniKontakt": {
                $kontaktID  = $_POST["kontaktID"];
                $ime        = $_POST["ime"];
                $prezime    = $_POST["prezime"];
                $adresa     = $_POST["adresa"];
```

```
                $telefon    = $_POST["telefon"];

                $sqlIzmena = 'UPDATE kontakti SET
                            Ime         = "'.$ime.'",
                            Prezime     = "'.$prezime.'",
                            Adresa      = "'.$adresa.'",
                            Telefon     = "'.$telefon.'"
                        WHERE
                            KontaktId   = "'.$kontaktID.'"';
                echo $sqlIzmena;
                mysql_query("SET NAMES utf8");
                if(!mysql_query($sqlIzmena)) {
                    die('Greska! Kontakt nije unet. Detalji greske: '.mysql_error());
                }
            }
            break;
        case "obrisiKontakt": {
                $idKontakta = $_POST["kontaktID"];

                $sqlObrisi = 'DELETE FROM kontakti WHERE KontaktId = "'.$idKontakta.'"';

                echo $sqlObrisi;
                mysql_query("SET NAMES utf8");
                if(!mysql_query($sqlObrisi)) {
                    die('Greska! Kontakt nije obrisan. Detalji greske: '.mysql_error());
                }
            }
            break;
    }
}
?>
```

[/CODE]

...ukratko, obzirom da se u ovom fajlu nalazi ceo kod koji aplikacija koristi, da se skripta ne bi uvek cela izvršavala, na osnovu parametra *actions* koji saljemo skripti u post-u preko ext-a definišemo koja akcija tacno treba da se izvrši (npr. akcija za ucitavanje kontakata, izmenu, brisanje...)

Rečnik pojmova

Namespace je apstraktni kontejner ili okruženje napravljen da čuva logičke grupe i jedinstvene identifikatore ili simbole.

Identifikator koji je definisan u namespace-u je asociran samo sa tim namespace-om. Istim identifikator može biti definisan u više različitih namespace-ova. To znači da značenje asocirano na identifikator u jednom namespace-u može ali i ne mora da ima isto značenje kao isti identifikator definisan u drugom namespace-u.

Jezik koji podržava namespace specifikuje pravila koja odredjuju kojem namespace-u identifikator pripada.

U programiranju scope je najprostije rečeno zatvaranje konteksta gde su vrednosti i izrazi asocirani. Različiti programski jezici imaju različite tipove scope-ova. Tip scope-a odredjuje koje vrste entiteta može da sadrži i kako se odražava na njih.

Uobicajno scope se koristi da se definiše vidljivost i doseg “sakrivenih” informacija.

Scope moze da:

- Sadrži deklaraciju ili definiciju identifikatora.
- Sadrži izraze koje definišu izvršavajući algoritam ili bar deo njega.
- Se ugnježdava ili da bude ugnjezden.

Events (Dogadjaji) - Dogadjaji su odazivi korisnikovih akcija ili promena stanja izvora dogadaja.

Neke definicije pojmova su preuzete sa wikipedije i mozda deluju nejasno, ali nisam umeo da ih sam objasnim bolje, ali ko želi u svakom slučaju na raspolaganju ima google pa može da potraži više informacija o pojmovima koji su mu nejasni.

Reč autora

Svako dobro svim čitaocima koji su preživeli ovaj dokument do kraja, koji ih nadam se nije previše smorio, i koji se možda nekom neće svideti i činiti se nedorečenim, ali, naravno, ukoliko neko bude imao neko pitanje može me slobodno kontaktirati ili dodati jos nešto svoje.

Vladica Savić
vladica.savic@gmail.com