BACKBONE BASICS

peepcode

- model
- view
- router

I - Basic

- model
- view
- collection
- router

II - App

- UI interaction
- Data manipulation
- Object designs

III Network

- save and load data
- form interactions
- alternate data stores

- u kodu od aplikacije je uključen web server
- napravit vim genset za projekt

↳ foreman server → za pokretanje servera

- localhost: 9292 (za vidit aplaciju)

- većina stvari se nalazi ili u app/public ili u app/test.js direktoriju

- javascript requirements
  - ↳ jquery → HTML DOM manipulation
  - ↳ underscore → korisne iteration fje i helper metode
  - ↳ backbone → mvc library

/app/public/index.html

/app/public/js/Tunes.js → access u jquery

Model

ALBUM MODEL

public/js/tunes.js

Aplikacijski model → Basic model ↳ dalje dodavat i extendat

```
(function ($) {

    Album = Backbone.Model.extend({});

}) (jQuery);

window.Album = Backbone.M....;
```

ovaj model je sad dostupan externally (u konzoli)

U DEVELOPER KONZOLI

```
> album = new Album ({title: 'Abbey Road', artist: 'The Beatles'})
```

GETTER i SETTER

get (key) set ({key: value})

isNew()
↳ vraća true ako model nije persistent (nije spremljen)

toJSON()
↳ vraća sve keys/values za model (za serijalizaciju)

```
> album.get('title')
        ↳ 'Abbey Road'
```

```
> album.isNew()
        ↳ true  (nije spremljen)
```

```
> album.set({title: 'Revolver'})
```

View

VIEW

tunes.js

```
...
window.AlbumView = Backbone.View.extend({

    initialize: function() {
        this.template = _.template( $('#album-template').html() );
    },

    render: function() {
        var renderedContent = this.template(this.model.toJSON());
        $(this.el).html(renderedContent);
        return this;
    }

});
```

ID iz index.html – za ispis titlea i artista

specificira template engine

OVO POPUNI TEMPLATE S VRIJEDNOSTIMA

jquery objekt

da se može chainat

VIEW.ELEMENT
.EL JE PROPERTY KOJEG BACKBONE DODA NA VIEW

render
↳ kombinira template i podatke (potrepći model za HTML templateom)
↳ appenda u DOM (appenda ili inserta u potrepći HTML)
↳ return 'this' (da se može raditi chaining)

VIEW – ZAŠTO TREBA BIT ZAKAČEN NA window OBJEKT ?

- instancier model
- instancier AlbumView
- put in a document and render

bonus

> album = new Album ({ title: 'Abbey Road', artist: 'The Beatles', tracks: [{ title: 'Track 1' }, { title: 'Track 2' }] });

↳ pour album key in a tracks

> albumView = new AlbumView ({ model: album });

↳ because every Album needs an albumView to render

> $ ('#container').append (albumView.render().el)

↳ add the album in the DOM

Model change events

MODEL CHANGE EVENTS
CSS / styling

DOM property - DODAVANJE CSS TAGOVA

↳ id - manualno setta id

↳ tagName - setta HTML tag, div, li, span

↳ className - setta css klasu

↳ el - overall element iz viewa

- u Tunes.js (na vrh AlbumView-a dodat)

```
tagName: 'li',
className: 'album',
```

- u browseru ponovit
  → album model
  → album view
  → #container append
  - i sad je content stiliziran sa css-om (negdje u projektu postoji gotovi css)
  - klasa i tag su generirani u view-u

MODEL CHANGE EVENTS

↳ change detection
- kad se editira objekt sa set → zove se change
  → notificirani se objekti (za slučaj view-a on se re-renderira)

- backbone je data driven
  ↳ umjesto viewa radi se sa podatcima
  ↳ view-i reagiraju na user actions by modifying data ~~a prosljeđuje~~ and change events cascade down
  ↳ ne događa se automatski

① Observe

View pita model da ga obavijesti kod se dogodi 'change' event
Registrira callback method

② React

Callback je notificirana kod su promijenjeni atributi modela

Tunes.js

Album View, initialize metoda, na vrh dodat

```
_.bindAll(this, 'render');
this.model.bind('change', this.render);
```

↑ event ↑ callback method

U browseru

setat album model, album view, appendat u #container

```
> album.set({artist: 'Soundgarden'})
```

→ čim se ovo setta, promijeni se artist u view-u

Collections

COLLECTIONS

↳ array of objects

u Tunes.js → collections u Plurals! models u singulars?

```
window.Albums = Backbone.Collection.extend({
    model: Album,        → collection zu koji model
    url: '/albums'
});
```

DEFINICIJA COLLECTIONA (model i url)

app/lib/server.rb

```
get "/albums" do
  content_type "application/json"
  File.readlines("public/albums.json")
end
```

ovo je kao baza

↳ u ovoj fileu se nalazi array of key:value objects (kao baza)
↳ u produkciji bi još postojao id

upotreba collectiona u konzoli

```
> albums = new Albums()
> albums.fetch()      → radi request na server i asinkrono se populira sa contentom
> albums.models       → daje objekte fetchane?
```

na collection se mogu zvat sve metode iz underscore.js

Collections
↳ each, map, select, reject, pluck, max, min, sortBy, size

Arrays
↳ first, rest, last, compact, flatten, without, union, intersection, uniq, indexOf, range

Functions
↳ bind, bindAll, memoize, delay, defer, throttle, once, after, wrap

PRIMJER

albums.map(function(album) { return album.get('title') })

↳ mića array titlova od albuma iz collectiona

↳ map - performa neku fju na svim objektima u colbction i mića array sa resultatima

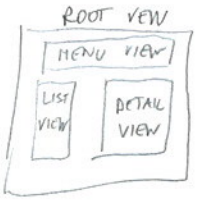
albums.pluck('title')

↳ dava single attribute od svih objekata

- collectioni imaju evente: add, remove, reset (za refresh viewa valjda)

RENDERIRANJE VIEWA ZA COLLECTIONSE

- u backboneu, neki viewi mogu imati druge viewe enkodane



Tunes.js

extenda AlbumView

```
window.LibraryAlbumView = AlbumView.extend({

window.LibraryView = Backbone.View.extend({
    initialize: function() {
        _.bindAll(this, 'render');
        this.template = _.template($('#library-template').html());
        this.collection.bind('reset', this.render);
    },
```

tagName: 'section',
className: 'library',

definira da je ovaj id template za ovaj view

BINDA DA JE ZA RENDER METODU THIS UVIJEK VIEW

ovaj template treba dodat u index.html

- slično kao album-template

```
render : function() {
    var $albums,
        collection = this.collection;
    $(this.el).html(this.template({}));
    $albums = this.$(".albums");
    collection.each(function(album) {
        var view = new LibraryAlbumView({
            model : album,
            collection : collection
        });
        $albums.append(view.render().el);
    });
    return this;
}
});
```

OVO BACKBONE SAM IZVLAČI IZ ARGUMENATA

PRAZAN OBJEKT

KOLEKCIJA NEMA SVOJE PODATKE

U BROWSER KONZOLI

```
> library = new Albums()
> libraryView = new LibraryView({ collection : library })
> $('#container').append( libraryView.render().el )
> library.fetch()
```

→ sad se ništa ne vidi

→ sad se dohvate albumi i prikažu

Albums.fetch()

↳ loads data from server, network request for JSON data

↳ fires 'reset' event when done

Albums.reset([{ title : '...' }])

↳ calls 'reset' directly with an array of key/value objects

↳ fires 'reset' event when done

ROUTER

Router

- router mepira fake-URL na actions-e, kreira evente kad su nule matchane

Tunes.js

```
window.library = new Albums();
```

public/index.html

```
<script type="application/javascript">
    jQuery(function() {
        window.library.fetch();
    });
</script>
```

kad se stranica učita, jQuery fetcha library-collection sa albuma

Tunes.js — ROUTER

```
window.BackboneTunes = Backbone.Router.extend({
    routes: {
        '': 'home'
    },
    initialize: function() {
        this.libraryView = new LibraryView({
            collection: window.library
        });
    },
    home: function() {
        var $container = $('#container');
        $container.empty();
        $container.append(this.libraryView.render().el);
    }
});
```

- routes: metoda koja se zove kad je url matchan
- initialize: ← setupira root view, ali ne renderira niti appenda u dokument
- home: ← ovdje se učitava view

ROUTE HANDLING METHOD

TUNES.JS – INSTANCIRANJE ROUTERA (kad se document loada)

```
$(function() {
    window.App = new BacklboneTunes();
    Backbone.history.start();
});
```

OVO JE "RANDOM" VARIJABLA, NIJE NEKI NAMESPACE

← PUSH STATE OPTION
/ umjesto # u url-u
- # je default
- { pushState: true }

SADA
↳ u browser možemo doć, ravno refreshat i bit će kontenta (ne treba inicirat model i to)

DODAVANJE NOVE RUTE

u routeru

```
routes: {
    '': 'home'
    'blank': 'blank'        ← (1) dodat ruti
},
;
blank: function() {         ← (2) dodat blank metodu
    $('#container').empty();        ← 2a) praznimo stranicu
    $('#container').text('blank');  ← 2b) dodajemo text "blank" kad se pozove blank route
}
```

- nakon ovoga posjetimo localhost:9292/#blank → i vidit ćemo text blank
- PROGRAMSKA NAVIGACIJA (u konzoli)

```
> App.navigate("blank", true)
```
↳ pozvat će se /#blank stranica

BACKBONE INTERACTIVITY II

- dodavanje buttona u "#album-template" (public/index.html)

```
<button class="queue add"><img src="/images/add.png" /></button>
<button class="queue remove"><img src="/images/remove.png" /></button>
```

- dodat event u LibraryAlbumView

EVENT HANDLER u

```
window.LibraryAlbumView = AlbumView.extend({
    events: {
        "click .queue.add": 'select'
    },
    select: function () {
        this.collection.trigger('select', this.model);
        console.log("Triggered select", this.model);
    }
})
```

→ OVO JE VIEW ZA JEDNU INSTANCU MODELA KOJA SE PRIKAZUJE U COLLECTIONVIEW

→ metoda

→ dodana da se 'select' metoda pozove kad netko klikne add button

EVENT

CSS SELEKTORI DA SE ZNA KOJEM ELEMENTU DODIJELITI EVENT

specifikacija metode

→ u ovoj metodi NE manipuliramo DOM elemente, nego upravljamo podatcima (koji onda mijenjaju view)

→ dodajemo console.log da vidimo da se nešto događa

mogu se koristiti standardni eventi iz jQuery libraryja

PLAYLIST COLLECTION

ROLES
↳ each collection may hold different data but use the same model

CUSTOM QUERIES
↳ server-backed collections may modify their URL to fetch different data, but use the same model to store it.

- napraviti subklasu od Albums collectiona da drži Playlist selection

```
// extendanje Album klase
window.Playlist = Albums.extend({
    isFirstAlbum: function(index) {
        return (index == 0)
    },
    isLastAlbum: function(index) {
        return (index == (this.models.length - 1))
    }
});
```

PLAYER MODEL

```
window.Player = Backbone.Model.extend({
    defaults: {
        'currentAlbumIndex': 0,
        'currentTrackIndex': 0,
        'state': 'stop'
    },
    initialize: function() {
        this.playlist = new Playlist();
    },
    play: function() {
        this.set({'state': 'play'});
    },
    pause: function() {
        this.set({'state': 'pause'});
    }
```

→ DEFAULTS PROPERTIES
↳ set key/values-a sa kojom će nova instanca krenuti
- Može se prepisat sa vrijednostima koje se passaju kao properties modela kad se instancira

JOŠ NA DRUGOJ STRANI

nastavak PLAYER MODELA

```
isPlaying : function() {
    return (this.get('state') == 'play');
},
isStopped : function(){
    return (!this.isPlaying());
},
currentAlbum: function() {
    return this.playlist.at(this.get('currentAlbumIndex'));
},
```

→ METODA NA COLLECTION SIMA
↳ uzima index i vraća objekt na tom indexu

```
currentTrackUrl: function(){
    var album = this.currentAlbum();
    return album.trackUrlAtIndex(this.get('currentTrackIndex'));
},

nextTrack : function () {
    var currentTrackIndex = this.get('currentTrackIndex'),
        currentAlbumIndex = this.get("currentAlbumIndex");
    if (this.currentAlbum().isLastTrack(currentTrackIndex)) {
        if (this.playlist.isLastAlbum(currentAlbumIndex)) {
            this.set({'currentAlbumIndex': 0});
            this.set({'currentTrackIndex': 0});
        } else {
            this.set({'currentAlbumIndex': currentAlbumIndex+1})
            this.set({'currentTrackIndex': 0});
        }
    } else {
        this.set({'currentAlbumIndex': currentAlbumIndex+1})
        this.set({'currentTrackIndex': 0});
    }
    this.logCurrentAlbumAndTrack();
},
```

prevTrack → slično kao nextTrack

IMS JOS

→

nadenk PLAYER.PROTO.A

```
logCurrentAlbumAndTrack: function() {
    console.log("Player " +
        this.get('currentAlbumIndex') + ':' +
        this.get('currentTrackIndex'), this);

}
});
```

INSTANCIRANJE PLAYERA

```
window.player = new Player();
```

PLAYLIST ALBUM VIEW

```
window.PlaylistAlbumView = Album.extend({});
```

PLAYLIST VIEW

dodavanje PlaylistViewa u Router

U ROUTER INITIALIZE FJI

```
this.playlistView = new PlaylistView({
    collection: window.player.playlist;
    player:     window.player,
    library:    window.library
});
```

U ROUTER HOME METODI (na kraj)

```
$container.append(this.playlistView.render().el);
```

```
window.PlaylistView = Backbone.View.extend({
    tagName:   'section',
    className: 'playlist',

    initialize: function() {
        _.bindAll(this, 'render');
        this.template = _.template($('#playl
        this.collection.bind('reset', this.render);
    },
```

ovo nije router

```
PlaylistView
render: function() {
    $(this.el).html(this.template
        (this.player.toJSON()));
    this.$('button.play').toggle(this.
        player.isStopped());
    this.$('button.pause').toggle(this.
        player.isPlaying());
    return this;
}
```

TREBA NAPRAVITI HTML TEMPLATE (public/index.html)

```
<script type="text/template" id="playlist-template">
    <h1>Playlist</h1>
    <nav>
        <button class="control play">Play</button>
                pause  Pause
                prev   Prev
                next   Next
    </nav>
    <ul class="albums"></ul>
</script>
```

4 BUTTONA / css je već napravljen u aplikaciji

ADD & REMOVE ALBUMS

u LibraryAlbumView-u imamo - click event preko select u dodaje
- select metoda triggera custom select event

a sad treba PlaylistView bindat na ovaj select event

BACKBONE – dodavanje playlist viewa – koraci

GLOBALNA VARIJABLA ↓ — EXTENDA SE IZ VIEWA ↓

```
MusicPlayer.PlaylistView = Backbone.View.extend({

    tagName: 'section',
    className: 'playlist',
```

} CIJELI VIEW ĆE BITI UNUTAR TAG-a SECTION KOJI IMA KLASU PLAYLIST (OVO JE OPCIONALNO)

```
    initialize: function () {
        _.bindAll(this, 'render');
```

- UVIJEK KAD SE IZVRŠAVA RENDER METODA, IZVRŠAVAT ĆE SE NA OVOM OBJEKTU
- OBAVEZNO

```
        this.template = _.template($('#playlist-template').html());
```

- ASSIGNA TEMPLATE ZA VIEW
- AKO JE POTREBNO, DODAT TEMPLATE U HTML DOKUMENT

```
        this.collection.bind('reset', this.render);
```

- this.collection JE KOLEKCIJA OBJEKATA KOJA SE PASS-a VIEW-u KAO PARAMETAR
- KAD SE DOGODI 'RESET' EVENT NA TOJ KOLEKCIJI, ~~REF~~ POZOVE SE RENDER METODA NA OVOM VIEWU

```
    },

    render: function () {
        $(this.el).html(this.template());

        return this;
    }

})
```

- U OVAJ VIEW POSTAVLJA TEMPLATE (KOJI JE PRIDRUŽEN U INITIALIZE)
- NIJE NUŽNO, ALI OMOGUĆUJE CHAINANJE

```
$(document).ready(function() {
    playlist = new MusicPlayer.Playlist();
    playlistView = new MusicPlayer.PlaylistView({collection: playlist});

    $('#playlist').append(playlistView.render().el);
```

- INSTANCIRANJE KOLEKCIJE
- INSTANCIRANJE VIEW-a + PASSANJE KOLEKCIJE KAO PARAMETAR

↳ UMETANJE VIEWA U HTML

PLAYLIST VIEW

PARAMETRI : { collection : playlist, library : library }

```
initialize : function () {
    ...
    this.library = this.options.library;
    this.library.bind ('select', this.queueTrack);
}
queueTrack: function (track) {
    this.collection.add (track);
}
```

- SAMO PRIDJELJUJE DA LIBRARY BUDE LIBRARY IZ PARAMETRA
- POZIVA METODU NA SELECT EVENT

- TRACK JE THIS.MODEL IZ TRACK.VIEWA
- DODAVA TRACK U KOLEKCIJU - DIŽE SE ADD EVENT NA COLLECTIONU

DODAT U INITIALIZE

```
this.collection.bind ('add', this.renderTrack);
```

- KAD SE DOGODI ADD EVENT NA COLLECTIONU ZOVE SE OVA METODA

```
renderTrack : function (track) {
    var view = new MusicPlayer.TrackView ({
        model : track,
        collection : this.collection
    });
    this.$('ul').append (view.render().el);
}
```

- U VARIJABLU VIEW SPREMA GENERIRANI TRACK VIEW
- DODAJE NEW TRACK U PLAYLIST VIEW

U PLAYLISTVIEW ne zaboravit

```
_.bindAll (this, 'render', 'renderTrack', 'queueTrack');
```

→ bindat nove metode na trenutni objekt

BACKBONE - dodavanje modela iz collectiona u collection

U OSNOVI

- track view diže event na kolekciji u kojoj se nalazi
  ↳ collection je passan view-u kao parametar u ovu svrhu
  → collection je library

- u playlist viewu imamo ~~collection view~~ library collection kao parametar. Bindamo da se izvrši neka metoda na event koji diže track view
  - metoda dodaje model u playlist collection, automatski diže 'add' event na tom collectionu

- u playlist view-u bindamo novu metodu na "add" event od collectiona. Ta metoda dodaje model na playlist view

---

TRACK VIEW

```
events: {
    'click .queue.add': 'select'
},
select: function () {
    this.collection.trigger('select', this.model);
    // console.log("I just triggered a select", this.model);
}
```

- ZOVE SELECT METODU KAD SE DOGODI CLICK NA .queue.add ELEMENTU

- TRACK VIEWU PASSANO COLLECTION KAO PARAMETAR - TAKO MU IMA PRISTUP
- DIŽE SELECT EVENT NA COLLECTIONU I KAO PARAMETAR ŠALJE MODEL