AngularJS

Kontroleri, Direktive, Filteri

AngularJS

- JavaScript framework za pravljenje klijentskih aplikacija
- Nastao je u Google, ali je sada open-source projekat
- https://angularjs.org/
- https://builtwith.angularjs.org/

Zbog čega uvodimo Angular

- Izgradnja većih aplikacija u JavaScript jeziku je prilično teška
  - prilagodljiv jezik, nedostatak provere tipova
  - koristi se u svim delovima – prilikom izgradnje korisničkog interfejsa, u komunikaciji između klijenta i servera, za implementaciju poslovne logike, za validaciju korisničkog unosa,...
  - ovo može rezultovati kodom koji se teško testira i koji je težak za održavanje

- Biblioteke kao što je jQuery mogu u značajnoj meri redukovati broj linija koda. Međutim, nedostaju im strukturne smernice koje bi pomogle prilikom razvoja velikih aplikacija

- Za ovo nam služe arhitekturalni šabloni (kao što je MVC, Model View Controller šablon) i radni okviri kao što je AngularJS

Dodavanje AngularJS-a u aplikaciju

- Jedini skript koji treba dodati u aplikaciju (za osnovnu funkcionalnost) je:

```
<script src="angular.js"></script>
```

- Nema spoljašnjih zavisnosti

- Takođe je neophodno dodati **ng-app** *direktivu*
  - najčešće se dodaje kao atribut u HTML tag (mada može biti dodata kao atribut u bilo koji HTML element)
  - Definiše se tačno jednom u aplikaciji
  - Govori Angularu da tretira sve unutar taga gde je ova direktiva dodata kao Angular aplikaciju i da ga u skladu sa time procesira
  - AngularJS izrazi mogu da se koriste u sekciji DOM stabla u kojoj je navedena ova direktiva

Dodavanje AngularJS-a u aplikaciju – primer

```
<html ng-app>
   <head>
        <script type="text/javascript" src="angular.js"></script>
   </head>
   <body>
        <h1>My first Angular example</h1>
        {{ 555 / 33 }}
   </body>
</html>
```

My first Angular example

16.818181818181817

Interpolacija: deo markup-a označen sa {{ *izraz* }} se zamenjuje vrednošću izraza navedenog unutar zagrada

```
<html >
   <head>
        <script type="text/javascript" src="angular.js"></script>
   </head>
   <body>
        <h1>My first Angular example</h1>
        {{ 555 / 33 }}
        <div ng-app > {{ 555 / 33 }} </div>
   </body>
</html>
```

My first Angular example

{{ 555 / 33 }}

16.818181818181817

Funkcionalnost AngularJS-a dostupna je samo u okviru taga u kome se nalazi ng-app direktiva

01_dodavanjeAngulara/index.html

Podešavanje servera

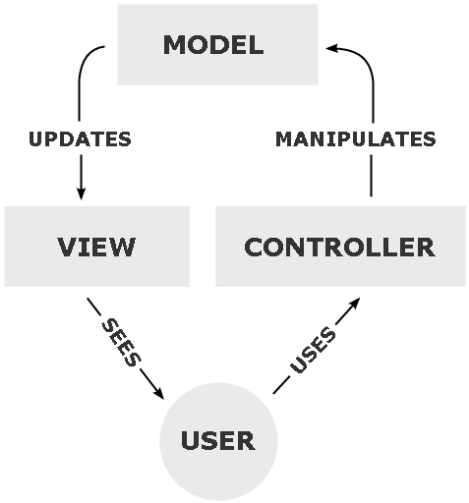
- Koristićemo http-server modul Node.js-a
- Instalacija servera:
  - Instalirati Node.js: http://nodejs.org
  - U komandnoj liniji ukucati:

    npm install http-server –g
  - Ovom komandom smo izvršili globalnu instalaciju HTTP servera

- Pokretanje servera:
  - ući u folder gde se nalazi kod aplikacije i uneti komandu:

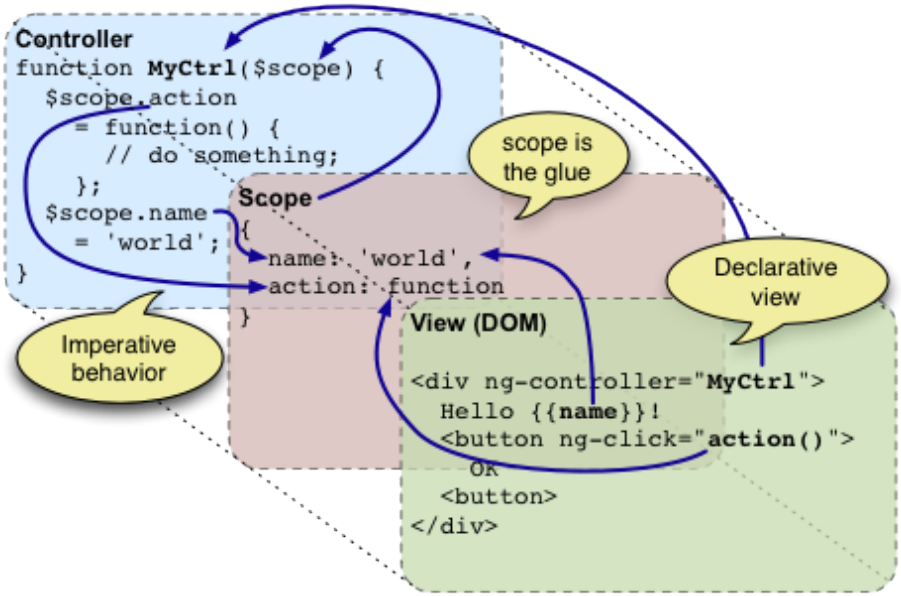
    http-server
  - Ovo pokreće server na adresi http://localhost:8080 koji može da servisira fajlove iz datog direktorijuma
  - U komandnoj liniji se pozicionirati na direktorijum 01_dodavanjeAngulara i pokrenuti http-server. U browser-u otići na adresu http://localhost:8080/ - otvoriće se stranica index.html iz datog direktorijuma

AngularJS – kontroler, scope i prikaz



Model View Controller (MVC) šablon

- Često se koristi u razvoju web aplikacija
- Tri komponente:
  - **Model**: čuvanje podataka aplikacije, poslovna logika i funkcije
  - **View (prikaz)**: reprezentuje model kroz korisnički interfejs
  - **Controller**: služi za koordinaciju modela i prikaza



AngularJS: Kontroler, scope, prikaz:

- **Kontroler**: manipulacija modelom
- **View (prikaz)**: model se prikazuje pomoću HTML-a
- **$scope**: veza sa modelom – svi objekti dodati u $scope predstavljaju model i mogu se predstaviti prikazom

Kontroleri

- Dodaju se u HTML stranice putem **ng-controller** *direktive*
- U AngularJS-u predstavljaju JavaScript funkcije u koje parametri mogu da budu prosleđeni pomoću *Dependency Injection mehanizma*

```
(function() {
   var HelloWorldController = function($scope) {
            $scope.message = "Hello, World!";
   };
   ...
})();
```

HelloWorldController.js

$scope je injektovan u kontroler

Kontroler je u suštini funkcija koja je dodeljena varijabli (U ovom slučaju HelloWorldController varijabli).

```
(function(){
    ...
})();
```

Ovo je JavaScript šablon IFFE (Immediately invoked function expression) – funkcija koja odmah poziva samu sebe. Na ovaj način se izbegavaju globalne varijable. Sve varijable definisane sa var su lokalne u okviru IFFE funkcije

```
<html ng-app=„...">
 <head>
  ...
  <script src="HelloWorldController.js"></script>
 </head>
 <body>
  <h1 ng-controller="HelloWorldController">
   message: {{message}}
  </h1>
 </body>
</html>
```

index.html

Pomoću ng-controller direktive specificiramo ime kontrolera koji kontroliše dati deo stranice

Sadržaj modela $scope.message

Šta se desi ako imamo štamparsku grešku, npr. {{ messag }} ?

Moduli

- Moduli obezbeđuju bolju organizaciju koda (bolji reusability i lakše testiranje)
- Kontrolere obično smeštamo u module. Ovim se izbegava globalni namespace
- Definicija modula:

angular je jedina globalna varijabla (deo Angular framework-a)

lista zavisnosti: modula koje naš modul uvozi

```
(function(){
   var app = angular.module("myFirstModule", []);
})();
```
app.js

- Dodavanje kontrolera u modul:

```
(function() {
     var HelloWorldController = function($scope) {...};

     var app = angular.module("myFirstModule");
     app.controller("HelloWorldController", HelloWorldController);
}());
```
HelloWorldController.js

Preuzimanje REFERENCE na prethodno kreirani modul, nije kreiranje modula (nema navedene liste zavisnosti)

Modul ima funkciju controller() kojom registrujemo funkciju HelloWorldController kao kontroler koji će biti dostupan pod imenom „HelloWorldController“

- Vezivanje modula za AngularJS aplikaciju:

Vrednost u ng-app direktivi govori Angularu da učita modul registrovan pod imenom „myFirstModule“

```
<html ng-app="myFirstModule">
```
index.html

02_kontroleri/index.html

Interpolacija {{ *izraz* }}

- Interpolacija : deo markup-a označen sa {{ *izraz* }} se zamenjuje vrednošću izraza navedenog unutar zagrada

- Interpolacioni izrazi se mogu staviti gotovo bilo gde u HTML-u:
  - **Unutar HTML taga** {{ *izraz* }}
  - **Vrednost atributa** <div att = {{ *izraz* }}>
  - **Ime atributa** <div {{ *izraz* }} = “vrednost_atributa“>

Izrazi

- Izraz se evaluira u skladu sa $scope objektom
  - U primeru: $scope objekat smo prosledili u funkciju kontrolera, a kontroler je zakačio message property za $scope
  - Interpolacioni izraz može da referencira property-je zakačene za $scope i poveže ih sa prikazom
  - **Promene property-ja modela se automatski sinhronizuju sa prikazom**

- U Angularu izraz koji nije definisan dobija vrednost undefined ili null
- Primeri izraza:
  - {{ *property* }} → vrednost $scope.*property*
  - {{ property >=0? ‘positive’, ‘negative’}} → ukoliko je ispunjen uslov prikazaće se ‘positive’, a ako nije ‘negative’
  - {{ *metoda()* }} → prikazaće se vrednost koju vraća metoda
  - {{ x = *metoda*() }} → ništa se neće prikazati, ali će $scope.x dobiti vrednost koju vraća metoda

Korišćenje kontrolera

- Kontroler upravlja modelom i prikazom
  - Kontroler **nikada** direktno ne manipuliše prikazom (HTML elementima)
  - Manipuliše **modelom** (kroz $scope) koji se vezuje za prikaz i ažurno stanje modela se prikazuje u browser-u
  - Prikaz se osvežava direktno iz modela, bez posredovanja kontrolera
  - Jedan kontroler može da ima više prikaza
  - Događaji u prikazu pokreću akcije kontrolera koje mogu
    - Da izmene model
    - Da korisnika preusmere na drugi prikaz

Više kontrolera u jednoj aplikaciji

```
(function() {
   var HelloWorldController = function($scope) {
        $scope.message = "Hello, World!";
   };
   var app = angular.module("myFirstModule");
   app.controller("HelloWorldController", HelloWorldController);
}());
```

HelloWorldController.js

```
(function() {
   var WeatherController= function($scope) {
        $scope.message = "The weather is cloudy";
   };
   var app = angular.module("myFirstModule");
   app.controller("WeatherController", WeatherController);
}());
```

WeatherController.js

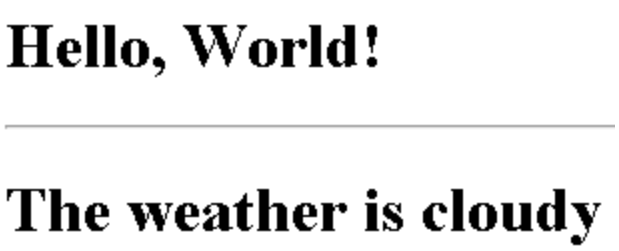
```
<div ng-controller="HelloWorldController">
    <h1> {{message}}</h1>
</div>
<div ng-controller="WeatherController">
    <h1> {{message}}</h1>
</div>
```

index.html

$scope.message iz HelloWorldController

$scope.message iz WeatherController

- Tipična situacija u razvoju aplikacija jeste da imamo više kontrolera



03_visestruki_kontroleri/index.html

Ugnježđeni kontroleri

```
<div ng-controller="ParentController">
    <h1> {{message}}</h1>
    <div ng-controller="ChildController">
        <h1> {{message}}</h1>
    </div>
</div>
```
index.html

$scope-ovi su hijerarhijski: varijable i funkcije koje su definisane u parent kontroleru su dostupne i u child kontroleru

```
$scope.message = "I am the parent";
```
ParentController.js

```
$scope.message nije definisan
```
ChildController.js

$scope.message iz parent $scope-a se kopira u child $scope

I am the parent

I am the parent

04_ugnjezdeni_kontroleri/01_inherit/index.html

```
$scope.message = "I am the parent";
```
ParentController.js

```
$scope.message = "I am the child";
```
ChildController.js

Ako je u $scope child-a definisan property istog imena kao u parent-u, u kontekstu child-a se biće korišćena vrednost iz child-a

I am the parent

I am the child

04_ugnjezdeni_kontroleri/02_redefine/index.html

Ugnježđeni kontroleri – nastavak

- Varijabla $parent se koristi za pristup $scope-u parent kontrolera
- Putem $parent child kontroleri mogu da modifikuju parent $scope. Ovo je jedan od načina da child kontroler obavesti parent kontroler da se nešto desilo

```
<div ng-controller="ParentController">
    <h1> {{message}}</h1>
    <p>{{text}}</p>

    <div ng-controller="ChildController">
     <h1> {{message}}</h1>
     <input type="text" ng-model="$parent.text"/>
    </div>
</div>
```

index.html

```
$scope.message = "I am the parent";
$scope.text = "Hello from the parent";
```

ParentController.js

```
$scope.message = "I am the child";
```

ChildController.js

Promena vrednosti u text box-u se sa child kontrolera propagira na parent kontroler

04_ugnjezdeni_kontroleri/03_accessParent/index.html

Složeni objekti

```
var WeatherController = function($scope) {
    $scope.weather={};
    $scope.weather.description = "The weather is cloudy";
    $scope.weather.temperature = "15 degrees Celsius";
};
```

WeatherController.js

Šta bi se desilo ako zakomentarišemo ovu liniju?

```
<h1 ng-controller="WeatherController">
    description: {{weather.description}} <br/>
    temperature: {{weather.temperature}} <br/>
</h1>
```

index.html

Šta bi se desilo da imamo štamparsku grešku, npr. weather.descriptio?

05_slozeni_objekti/index.html

Direktive

- U AngularJS-u su konstrukcije koje omogućavaju da se proširi standardni HTML rečnik
- Omogućavaju indirektnu interakciju prikaza sa $scope. **Dvosmerno mapiranje (two-way data binding)**: ako se promene podaci u $scope, automatski se ažurira prikaz; ako se promene podaci u prikazu automatski će se ažurirati $scope

- Do sada smo videli
  - **ng-app** – govori Angularu da tretira sve unutar taga gde je ova direktiva dodata kao Angular aplikaciju i da ga u skladu sa time procesira
  - **ng-controller** – povezuje kontroler sa prikazom
  - **{{ }}** – direktiva za vezivanje podataka (data-binding directive)

ng-model

- **ng-model**
  - pomoću ove direktive vrši se prenos podataka iz prikaza u model

```
(function() {
    var HelloWorldController = function($scope) {
        $scope.message = "Hello, World!";
    };
    ...
}());
```

HelloWorldController.js

```
...
<div ng-controller="HelloWorldController">
    <h1> {{message}}</h1>
    <input type="text" ng-model="message">
</div>
 ...
```

index.html

Inicijalna vrednost $scope.message koja se prikazuje prilikom otvaranja strane

Vrednost ng-model direktive je naziv property-ja koji će se dodati u $scope. U ovom slučaju će se vrednost koju korisnik ukuca u tekstualno polje postaviti u $scope.message.
(napomena: nije neophodno da message bude inicijalizovan u kontroleru, message property bi se svakako zakačio na scope)

Interpolacija se automatski ažurira prema vrednosti $scope.message

06_ng-model/index.html

ng-click i ng-submit

- **ng-click**
  - Vrednošću zadatom u ng-click direktivi unutar nekog elementa se zadaje funkcija koja će se pozvati kada korisnik klikne na taj element
- **ng-submit**
  - Slično kao ng-click, ali okida submit događaj u browseru umesto click događaja nad elementom

07_ng-click_ng-submit/index.html

ng-show i ng-hide

- Prikazivanje/skrivanje HTML elemenata bazirano na vrednosti zadatoj u direktivi:

  ** ** ili ** **

  Element će biti sakriven ukoliko je hidden==true

- Šta sve može da bude true u JavaScript-u (thruthy)? Sledeće vrednosti su uvek falsy:
  - false
  - 0 (nula)
  - “” (prazan string)
  - null
  - undefined
  - NaN (Not a Number)
  - Sve ostale vrednosti su thruthy

- Element koji se ne prikazuje je i dalje na stranici, samo je sakriven

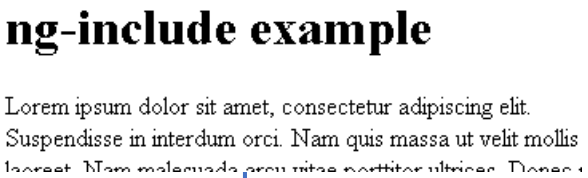
08_ng-show_ng-hide/index.html

ng-include

- Ugrađivanje HTML-a iz drugog izvora (npr. drugog HTML fajla) u prikaz
- Ovo je zgodno ukoliko želimo da razbijemo kompleksne stranice na razumljivije delove ili ukoliko imamo deo koji se može reuse-ovati

```
<html ng-app>
 <head>
  <script src="angular.js"></script>
 </head>
 <body>
  <h1>ng-include example</h1>
  <div ng-include="'LoremIpsum.html'"></div>
 </body>
</html>
```

index.html



Obratiti pažnju da je LoremIpsum.html pod jednostrukim navodnicima. U suprotnom bi Angular pokušao ovo da evaluira kao da je u pitanju izraz (jednostrukim navodnicima naglašavamo da je u pitanju string)

09_ng-include/index.html

ng-repeat

- Omogućava prolazak kroz kolekciju objekata

```
var EmployeesController = function($scope)
    $scope.data = {
        "employees":[
            {"firstName" : "Petar",
            "lastName" : "Petrovic",
            "salary" : "100"},
            {"firstName" : "Nikola",
            "lastName" : "Nikolic",
            "salary" : "300"},
            {"firstName" : "Marko",
            "lastName" : "Markovic",
            "salary" : "200"}
        ]};
};
```

EmployeesController.html

```
...                                    $scope.data.employees
<tbody>                                          ↓
    <tr ng-repeat="empl in data.employees">
        <td>{{ empl.firstName }}</td>
        <td>{{ empl.lastName }}</td>
        <td>{{ empl.salary }}</td>
    </tr>
</tbody>
...
```

index.html

10_ng-repeat/index.html

ng-class

- Omogućava dinamičku postavku CSS klasa HTML elemenata

  <ANY class="ng-class: expression;"> ... </ANY>

- Izraz može biti:
  - String sa imenima klasa odvojenim razmacima

    ng-class="text-center"
  - Niz class vrednosti

    ng-class="[text-center, text-danger]"
  - Izraz – mapa (objekat) gde su ključevi imena klasa, a vrednosti boolean (primeniti klasu ili ne)

    ng-class="{ 'text-success': txSuccess, 'bg-success': bgSuccess}"

    true/false izrazi

11_ng-class/index.html

Direktive – zaključak

- Direktive za vezivanje podataka
  - {{ *message* }}
- Direktive za rukovanje modelom
  - ng-model
- Direktive za rukovanje događajima
  - ng-click, ng-submit
- Direktive za rukovanje prikazom
  - ng-show, ng-hide, ng-include

- Videli smo samo mali deo skupa Angular direktiva (samo u osnovnoj biblioteci – Angularu ih ima preko 50)

Filteri

- Formatiranje podataka za prikaz (bez izmene samih podataka)

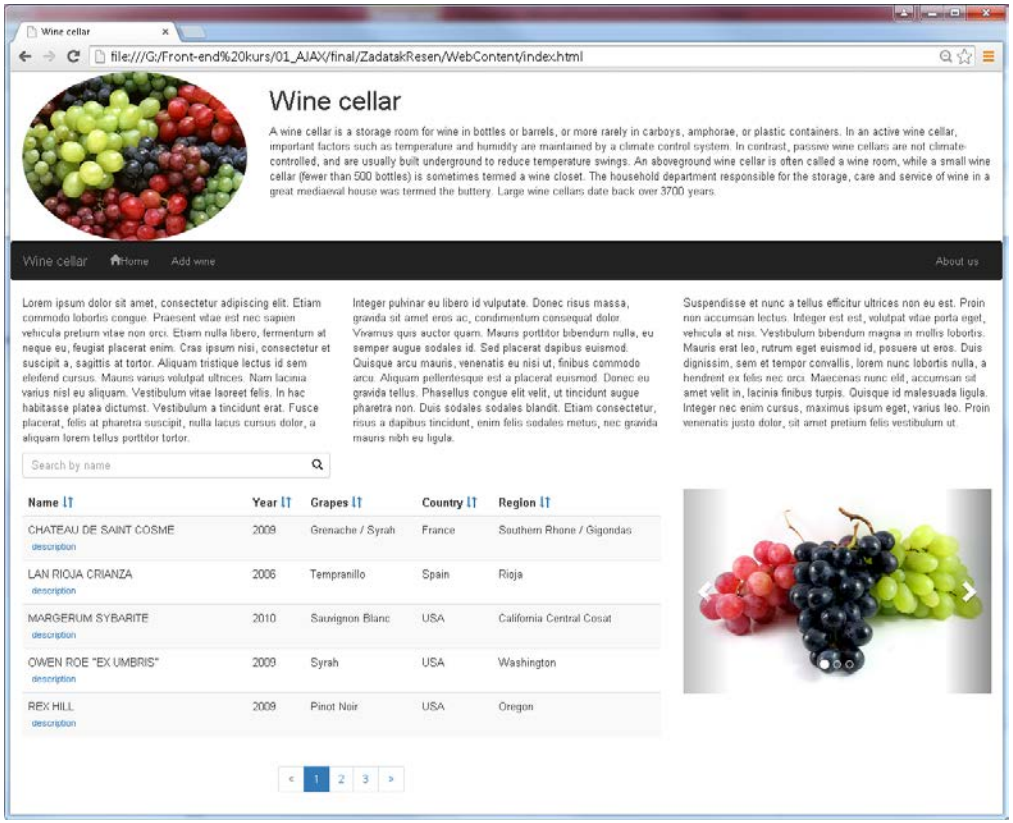
experssion|filterName:parameter

Filter	Namena	Primer
amount | currency[:symbol]	Formatira broj u valutu	1500| currency:"USD$"
date | date[:format]	Formatira datum u string po formatu	1288323623006 | date:'yyyy-MM-dd HH:mm:ss Z'
array | filter:expression	Podniz koji zadovoljava izraz. Izraz može biti string, objekat ili funkcija	repo in repos | filter:searchTerm
data | json	Konverzija JSON u string	{'name':'value'} | json
array | limitTo:limit	Podniz sa zadatim brojem elemenata	[1,2,3,4,5] | limitTo:3
string | lowercase	Pretvara string u mala slova	"AbCd"| lowercase
string | uppercase	Pretvara string u velika slova	"AbCd"| uppercase
number | number[:fractionSize]	Formatira broj u tekst	123123.45678 | number:4
array | orderBy:predicate[:reverse]	Sortira niz. Predikat može da bude funkcija, string ili niz. Reverse je boolean vrednost.	repo in repos | filter:searchTerm | orderBy:'name'

12_filteri/index.html

Wine cellar

- Zadatak: korišćenjem AngularJS-a i Bootstrap-a napraviti stranicu za pregled vina



13_wineCellar/index.html

Wine cellar

1. Napraviti stranicu index.html
   - uključiti AngularJS biblioteku
   - uključiti Bootstrap biblioteku
2. Napraviti modul winesModule i vezati ga za aplikaciju
3. Statički sadržaj:
   - Napraviti header stranice (Header.html) i uključiti ga u index.html pomoću ng-include
   - Napraviti navigacioni bar (Navbar.html) i uključiti ga u index.html pomoću ng-include
     - Da bi se navigacioni bar fiksirao na vrh index.html stranice ugnježden je u element sa data-spy="affix„ data-offset-top="280"
   - Napraviti tekst sa 3 kolone (LoremIpsum.html) i uključiti ga u index.html

Wine cellar - Navigacija

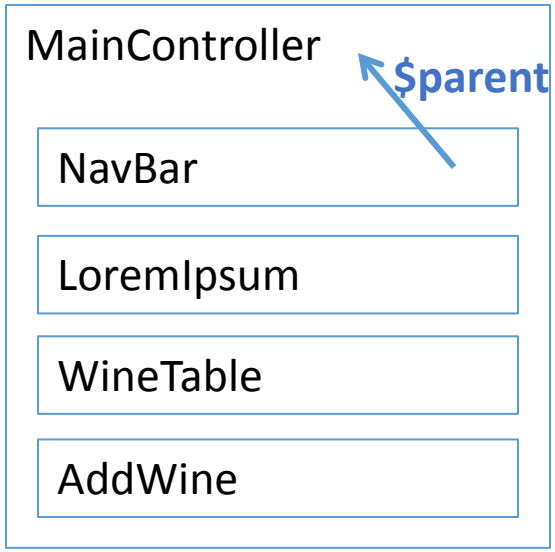
- Napravljene su dve HTML stranice:
  - AddWine.html koja će sadržati HTML elemente za prikaz forme za unos vina
  - WineTable.html koja će sadržati HTML elemente za prikaz i pretragu tabele sa vinima
- Napravljen je MainController koji u $scope ima dva property-ja:

  $scope.showWineTable = true; → inicijalno prikaži tabelu sa vinima

  $scope.showAddWine = false; → inicijalno sakri formu za prikaz vina
- MainController je postavljen tako da su u njega ugnježdeni tagovi u kojima se nalaze navbar , wineTable i addWine (MainController je parent)
- Privid navigacije je realizovan pomoću ng-show direktive:

Wine cellar - Navigacija

- Želeli bi smo da se korisnik prebacuje sa strane na stranu klikom na linkove iz navigacionog bar-a. Navigacioni bar je uključen pomoću ng-include u okviru dela stranice koji kontroliše MainController



ng-include kreira novi child scope koji nasleđuje $scope iz MainController-a

Pristup $scope.showAddWine u MainController-u

<a href="#" ng-click="**$parent**.showAddWine=false…">...

Wine cellar – učitavanje podataka

- Napravljen je WinesControler koji će da služi za kontrolisanje prikaza tabele sa vinima (WineTable.html)

- U wineData.json se nalazi JSON objekat sa podacima o vinima. Uneto ovaj objekat u WinesController:

```
var data = {
    "wines":[
            ...
    ]};
$scope.wines = data.wines;
```

Wine cellar

- Uz pomoć ng-repeat direktive u WineTable.html napravljena je tabela sa vinima

```
<tbody>
    <tr ng-repeat="wine in wines”>
        <td>
            <div>{{ wine.name }}</div>
            <button type="button" class="btn btn-link" data-toggle="collapse" data-
                    target="#{{wine.id}}">Show details</button>
            <div class="collapse" id="{{wine.id}}">
                <div><img ng-src="images/{{ wine.picture }}" height="100" /></div>
                <div>{{ wine.description }}</div>
            </div>
        </td>
        <td>{{ wine.year }}</td>
        ...
    </tr>
</tbody>
```

ng-src umesto src. U suprotnom bi browser pre nego što Angular stigne da evaluira {{...}} taj string protumačio kao URL i vratio grešku 404 (image bi se ipak prikazao ali bi greška ostala u konzoli). Sa ng-src se tek kada Angular evaluira binding izraz postavi src atribut na vrednost izraza.

- Napravljena je forma za unos vina u AddWine.html

Wine cellar - sortiranje

- Inicijalno je isključeno sortiranje i smer sortiranja je podešen na ascending:

```
$scope.sortCriteria = "";
$scope.sortDirection = "+";
```

- Definicija glyphicons kao linka:

```
<a href="#" ng-click="setSortCriteria('name')">
<a href="#" ng-click="setSortCriteria('year')">
...
```

- Funkcija $scope.setSortCriteria = function(newCriteria) postavlja $scope.sortCriteria na prosleđenu vrednost (name, year, ...) i/ili menja $scope.sortDirection

- Tabela:

```
<tr ng-repeat="wine in wines | orderBy: sortDirection + sortCriteria" >
```

Wine cellar – Pretraga po imenu

- Inicijalno je isključena pretraga po imenu: $scope.searchQuery="";

- Tekstualno polje:

  <input id="searchName" type="text" … ng-model="searchQuery">

- Tabela:

  <tr ng-repeat="… | filter: searchWine " >

Korisnički unos (kriterijum pretrage) će biti smešten u **$searchQuery** koji se koristi u funkciji searchWine

- Funkcija searchWine prima jedan red tabele (u ovom slučaju jedan wine JSON objekat) i vraca
  - true ukoliko red odgovara kriterijumu pretrage i treba da bude prikazan u tabeli
  - false ukoliko red ne odgovara kriterijumu pretrage i ne treba da bude prikazan u tabeli

```
$scope.searchWine = function(tableRowData){
    var wineName = tableRowData.name.toLowerCase();
    var query = $scope.searchQuery.toLowerCase();
    return wineName.indexOf(query) != -1;
}
```

Wine cellar - Paginacija

- Inicijalno $scope.maxWinesInTable =5; – prikazati max. 5 vina u tabeli
- U WinesController je izračunato:
  - $scope.noPaginationPages – koliko ima stranica paginacije
  - $scope.paginationPagesArr – kreiran je niz stringova koji će biti prikazan u komponenti za paginaciju, npr. ukoliko je $scope.noPaginationPages == 3, $scope.paginationPagesArr = [“<”,”1”,”2”,”3”,”>”]
  - $scope.activePage je trenutno aktivna stranica (inicijalno 1)
- Komponenta za paginaciju:

Za svaki String i iz niza $scope.paginationPagesArr napraviti po jedan unutrašnji <li> element

```
<ul class="pagination" ng-repeat="i in paginationPagesArr">
  <li ng-class="{'active': i == activePage}"><span ng-click="changePage(i)">{{ i }}</span></li>
</ul>
```

** tagu se postavlja klasa active ukoliko je String i jednak $scope.activePage**

kada korisnik klikne na stranicu u paginaciji poziva se funkcija $scope.changePage kojoj se prosleđuje i (String koji pokazuje na koju je stranicu kliknuto) i koja ažurira $scope.activePage

- Tabela:

```
<tr ng-repeat="restaurant in restaurants ... | limitTo : maxWinesInTable: (activePage-1)*maxWinesInTable">
```

WineCellar – dodavanje vina

- Napravljen je AddWineController koji kontroliše stranicu AddWine.html

- $scope.wines se nalazi u WinesController, a nama su ti podaci potrebni u AddWineController-u koji je sestrinski kontroler. Kako preneti podatke između kontrolera koji nisu ugnježdeni?
  - Kasnije će biti reči o „Controller as“ sintaksi
  - Za sada ćemo prebaciti podatke u MainController

- Svakom od input elemenata je dodeljena ng-model direktiva kako bi se odgovarajuća vrednost prenela u $scope
  - Svi unosi su enkapsulirani u jedan objekat newWine (npr. za name input: ng-model=newWine.name)

- Elementu <form> je dodata ng-submit direktiva koja poziva funkciju addWine iz AddWineController-a:

```
$scope.addWine = function(){
    $scope.newWine.id = $scope.newWine.name.split(" ").join("_");
    $scope. $parent.$parent.wines.push($scope.newWine);
}
```

WineCellar – redirekcija nakon dodavanja

- U addWine funkciju dodati:

```
$scope.newWine = {};
$scope.$parent.showWineTable = true;
$scope.$parent.showAddWine = false;
```

- Šta se dešava? AngularJS će često napraviti child scope u cilju da ima svoj key/value parove koji se ne mešaju sa podacima višeg nivoa. U ovom slučaju dodat je još jedan child scope - $parent.showWineTable menja showWineTable u tom child scope-u koji sakriva vrednost $scope. showWineTable iz MainController-a. Problem možemo rešiti sa:

```
$scope.$parent.$parent.showWineTable = true;
$scope.$parent.$parent.showAddWine = false;
```

WineCellar – redirekcija nakon dodavanja 2. način

- Generalno, ako želimo pristup gornjem scope-u bolja praksa je da se napravi objekat

```
$scope.showWineTable = true;
$scope.showAddWine = false;
```
MainController.html

Jednom kada se u child $scope-u napravi property showWineTable, taj property će „prekriti“showWineTable parent kontrolera

```
$scope.showPanels = {
        showWineTable: true,
        showAddWine: false
}
```
MainController.html

```
$scope.showPanels.showWineTable = true;
$scope.showPanels.showAddWine = false;
```
AddWineController.html

Podaci se ne čuvaju direktno na $scope: $scope.showPanels.showWineTable će prvo da izvrši čitanje (pronalaženje objekta) a tek onda pisanje

- Iz istog razloga je bolja praksa da se u ng-model direktivi stvari ne stavljaju direktno na $scope već enkapsuliraju u objekat (umesto ng-model=name stavili smo ng-model =newWine.name)
- Kasnije ćemo videti “Controller as” sintaksu kojom se elegantno rešava opisani problem

Zadatak

- U fajlu restaurantData.json je dat JSON objekat koji predstavlja restorane. Po ugledu na primer Wine cellar napraviti web aplikaciju za pregled restorana:
  - Napraviti navigacioni bar koji je fiksiran na vrhu stranice i koji se sažima na manjim prikazima
  - Deo HTML-a koji predstavlja navigacioni bar izmestiti na posebnu stranicu Navbar.html
  - Putem navigation bar-a obezbediti navigaciju između dve stranice
    - AddRestaurant.html, koja sadrži formu za dodavanje novog restorana
    - RestaurantTable.html, koja tabelarno prikazuje postojeće restorane

Zadatak

- RestaurantTable.html
  - Prikazati restorane u tabeli
  - obezbediti sortiranje u oba smera tabele po kriterijumima name,rating, price i cuisine
  - obezbediti paginaciju. Dodati i dugmad “<<” i “>>” koja vode na prvu, odnosno poslednju, stranicu tabele
  - svaki red tabele treba da sadrži i “Remove” dugme kojim se selektovani restoran briše
  - obezbediti pretragu prema željenom kriterijumu. Pored inputa za pretragu dodati select koji omogućava korisniku da vrši pretragu prema imenu restorana, rating-u, ceni ili vrsti hrane u zavisnosti od toga koji je kriterijum selektovan

- AddRestaurants.html
  - obezbediti unos novog restorana (dovoljno je uneti kriterijume iz tabele – name, rating, price, cuisine, working hours, location i description)
  - voditi računa da sva polja budu popunjena pre pokušaja unosa restorana
  - obezbediti redirekciju na stranicu RestaurantsTable nakon uspešnog unosa