# Razvoj korisničkog sučelja tehnologijom AngularJS

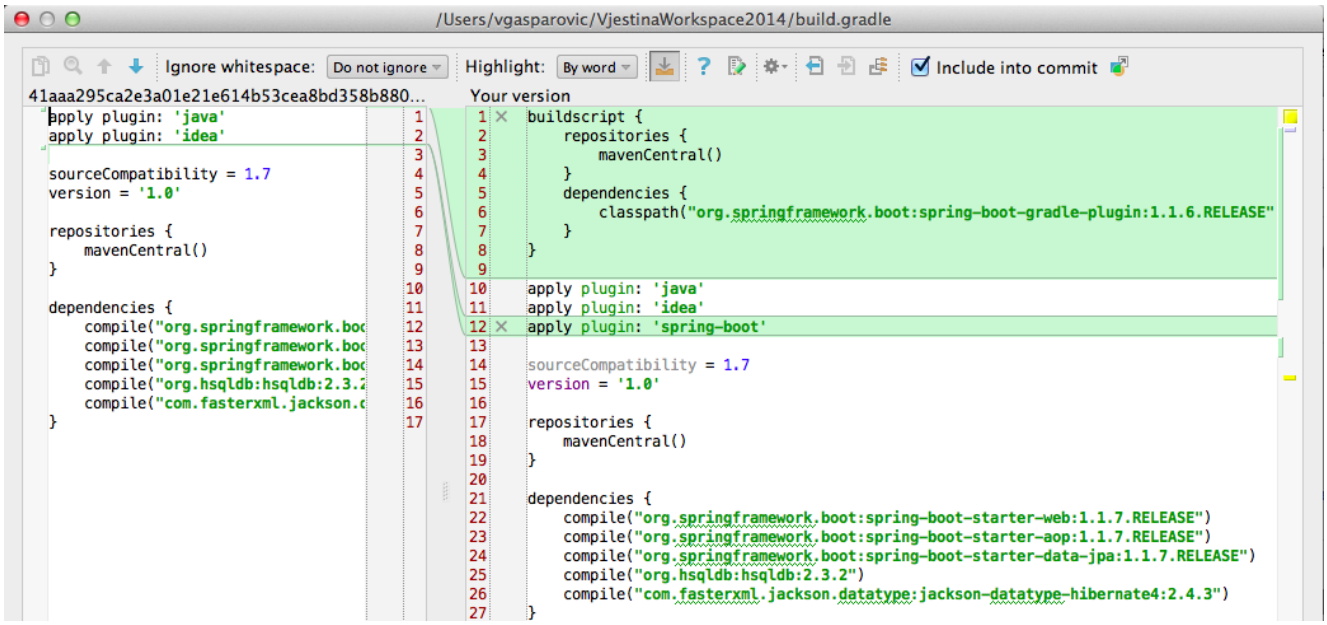
**Uvod**

Ovo predavanja zamišljeno je kao uvod u Angular.js, frontend tehnologiju koju ćemo koristiti za izgraditi korisničko sučelje za našu web aplikaciju. Proći ćemo kroz koncepte kao što su MVC, two way databinding, templates, DI, directives (extend html) i dr.

## build.gradle

Kako bi si ubrzali razvojni proces, dodati ćemo u build.gradle plugin za spring boot koji će nam dodati *task* imena bootRun koji će prilikom pokretanja kopirati sve statičke datoteke (iz direktorija src/main/resources), pa ćemo moći testirati frontend funkcionalnosti bez da svaki puta restartamo backend.

U nastavku prilažem screenshot razlika prošle i nove verzije build.gradle datoteke:




**Setup**

Statičke datoteke naše aplikacije stavljat ćemo u src/main/resources/public direktorij našeg projekta jer Spring Boot automatski sadržaj tog direktorija učini dostupnim preko weba.

**Dodajmo podršku za Angular.js**

Razvoj frontenda će se kompletno odvijati u paketu src/main/resources/public, pa ćemo krenuti s time da dodamo podršku za Angular dodavši u index.html sljedeće linije:

<script src="//ajax.googleapis.com/ajax/libs/angularjs/1.3.3/angular.js"></script>

<script src="//ajax.googleapis.com/ajax/libs/angularjs/1.3.3/angular-route.js"></script>

**Aplikacija**
Svrha ove AngularJS aplikacije je da omogućimo korisniku interakciju s backendom koji smo složili u sklopu prijašnjih predavanja.

Najprije definiramo početnu stranicu index.html koja će biti korijen svim drugim View u aplikaciji.

```
<html ng-app="tournamentApp">
<body>

<div ng-view></div
<script
src="//ajax.googleapis.com/ajax/libs/angularjs/1.3.3/angular.js"></script>
<script
src="//ajax.googleapis.com/ajax/libs/angularjs/1.3.3/angular-route.js"></script>

</body>
</html>
```

Primijetite da smo dodali neke atribute u <html> i  koji ne postoje u standardnom HTMLu. Ti atributi su Angular direktive koje su obično i prefiksirane s “ng-”. U našem slučaju iskoristili smo direktive *ng-app* i *ng-view*.

*Ng-app* direktiva je osnovna direktiva angulara i koristi se za bootstrap angular aplikacije. S tom direktivom Angularu “kažemo” da sve unutar tag-a gdje smo ju naveli dio Angular aplikacije koja je sadržana u modulu koji smo predali kao parametar (u našem slučaju “tournamentApp”).

Moduli u angularu su način grupiranja koda pod jednim imenom, slično kao paket (*package*) u Javi. Svaki modul je zapravo zasebna angular aplikacija koja može ovisiti o drugim modulima koje navedemo u njegovoj deklaraciji. Svaki modul može imati svoje kontrolere, servise, direktive, konfiguraciju, filtere, providere i dr.

*Ng-view* direktiva služi Angular-routing dijelu framework-a i ona govori angular na kojem mjestu u DOM strukturi treba ubaciti trenutačno aktivan View.

Idemo napraviti našu aplikaciju. U app.js dodamo ovaj kod i dodamo app.js u <script> od index.html.

```
angular.module('tournamentControllers', []);
angular.module('tournamentServices', []);

var tournamentApp = angular.module('tournamentApp', ['ngRoute',
  'tournamentControllers',
  'tournamentServices']);
```

S ovim smo stvorili module “tournamentController”, “touranemntServices, “tournamentApp” i naveli da modul “tournamentApp” ima “ngRoute” (angular-route modul), sa “tournamentControllers” i “tournamentServices” modulima kao ovisnostima (*dependencies*). Moduli se stvaraju naredbom angular.module(‘<ime modula>’, [<dependency list>]);, a dohvaćaju naredbom angular.module(‘<ime modula>’).

S ovime je završena osnovna konfiguracija aplikacije i možemo krenuti razvijati View-ove koje želimo prikazati korisniku, uz pripadajuće kontrolere i modele.

Kao što smo prije rekli angular će ubacivati View tamo gdje smo postavili <div ng-view>, a to će raditi u ovisnosti koji URL imamo u browseru. Pa idemo napraviti View za igrače na URL-u /players.

Prvo trebamo reći angularu da kad je na URL players da ubaci view koji želimo i da na njega poveže kontroller koji želimo. Tu konfiguraciju ćemo napraviti na glavnom modulu “tournamentApp” koji smo bili spremili u varijablu tournamentApp nakon deklaracije.

```
tournamentApp.config(function($routeProvider) {
  $routeProvider
    .when('/players', {
      templateUrl: 'partials/player-list.html',
      controller: 'PlayerListCtrl',
      resolve: {
        players: function (PlayerFactory) {
          return PlayerFactory.getPlayers();
        }
      }
    })
});
```

*Config* funkciji modula predajemo funkciju u kojoj ćemo konfigurirati rute u našoj aplikaciji. Na ovom mjestu koristimo angularov *Dependency Injetion* mehanizam, tako što napišemo $routeProvider i angular će onda u svim našim referenciranim modulima (uključujući i module frameworka) probati pronaći takav angular objekt i ubaciti na

mjesto tog parametra. U config funkciji ne možemo koristiti bilo koje objekte već samo "provider" objekte jer se ona izvodi prije početka aplikacije.

Na ovaj smo način angularu "rekli" da kada je URL "/players", template od View-a je na URL-u "partials/player-list.html" (postoji i property template, pomoću kojeg možemo direktno napisati template HTML), kontroler za ovaj View se zove "PlayerListCtrl", a sa *resolve* kažemo da prije nego što prikaže View dovrši sve funkcije koje smo predali kao property tog objekta što će u našem slučaju biti dohvat igrača (pri čemu će property resolve objekta biti dostupni u PlayerListCtrl kontroleru).

Napravimo sad kontroller za taj *view*.

```
var tournamentControllers = angular.module('tournamentControllers');

tournamentControllers.controller('PlayerListCtrl', function ($scope, PlayerFactory,
players) {
  $scope.players = players.data;

  $scope.deletePlayer = function (player) {
     PlayerFactory.deletePlayer(player.id)
        .success(function (data) {
           PlayerFactory.getPlayers()
              .success(function (data) {
                 $scope.players = data;
              })
        })
        .error(function (error) {
        })
  }
});
```

Na početku smo dohvatili postojeći modul "tournamentControllers" koji smo ranije deklarirali unutar *app.js* i dodali kao dependency glavnom modulu, pa na njemu deklarirali kontroller koji se zove "PlayerListCtrl". Funkcionalnost mu je sadržana u funkciji koju smo predali kao drugi parametar. Tu smo opet koristili DI od angulara da dobijemo $scope (ugrađeni servis), PlayerFactory - servis koji smo mi deklarirali, kasnije će biti objašnjen.

U kontroleru smo u $scope.players stavili podatke o igračima koje smo dobili sa servera. Objekt $scope predstavlja most (okolina/doseg) između kontrolera i viewa. Sve što se promijeni na tom objektu biti će vidljivo i iz view-a i iz kontrolera (neovisno o tome gdje

se promijenilo). Također smo u kontroleru na $scope definirali i metodu za brisanje igrača koja kao argument prima igrača za brisanje i nakon uspješnog brisanja dohvati ponovo igrače sa servera i postavi ih u $scope.players.

Sada idemo napraviti template iz kojeg će se izgraditi *view* za listu igrača.

```
<div>
  <h3>Player</h3>
  <table border="1">
    <tr>
      <th>Name</th>
      <th>Nickname</th>
      <th>Race</th>
      <th>Actions</th>
    </tr>
    <tr ng-repeat="player in players">
      <td>{{player.name}}</td>
      <td>{{player.nickname}}</td>
      <td>{{player.race}}</td>
      <td><a href="#/players/{{player.id}}">Edit</a> <a href
ng-click="deletePlayer(player)">Delete</a></td>
    </tr>
  </table>
  <a href="#/players-add-player">Add new player</a>
</div>
```

Prvo što će te primjetiti je korištenje {{}} zagrada. To služi za označavanje angular izraza koje treba evaluirati. U slučaju {{player.name}} angular će pokušati na tom mjestu prikazati sadržaj varijable player.name koja se nalazi unutar scope objekta. Ovdje smo iskoristili i direktivu *ng-repeat* koja će za svaki element u predanoj listi generirati HTML nad kojim je deklarirana, u našem slučaju za svakog igrača ponoviti redak u tablici s pripadajućim angular izrazima. Ova direktiva također kreira i novi scope za svako ponavljanje i postavlja odgovarajući player objekt u nju.
Na kraju još imamo direktivu *ng-click* koja kaže da se priliom klika na taj element pozove funkcija s danim argumentima. No kako je moguće da ta direktiva uopće vidi tu funkciju ako smo rekli da ng-repeat generira novi scope za svaki element (u kojoj onda vidimo taj player objekt koji predajemo)? To je zato što iako ng-repeat direktiva kreira novi scope, taj scope također nasljeđuje i parent scope u ovom slučaju je to scope od viewa gdje je definirana ta funkcija.

Znači iz ovog templatea će se generirati View s tabličnim prikazom podataka svih igrača i opcijom za brisanje pojedinog igrača te linkom na URL gdje ćemo implementirati dodavanje igrača.

Još trebamo definirati naš PlayerFactory objekt koji će nam služiti za obavljanje CRUD metoda vezanim uz igrače.

```
var tournamentServices = angular.module('tournamentServices');

tournamentServices.factory('PlayerFactory', function ($http) {
  var factory = {};
  factory.getPlayers = function () {
    return $http.get('players');
  };
  factory.deletePlayer = function (id) {
    return $http.delete('players/' + id);
  }
  return factory;
});
```

Factory objekti modula nam služe za izdvajanje poslovne logike i kompliciranijih operacija iz kontrolera, te povećavanju modularnosti naše aplikacije. U našem slučaju imamo samo dvije metode za dohvat svih igrača i brisanje igrača koje koriste $http objekt (koji definiran i injektiran od stranje angulara), koji služi za slanje HTTP upita.