We make IT

JCommerce

Development aplikacji frontendowej z AngularJS

Katowice, 21 marca 2016r.

- Organizacja struktur projektu
  - ✓struktura plikowa projektu
  - ✓struktura kodu aplikacji
- Elementy AngularJS
  - ✓Routing
  - ✓Controllers
  - ✓Directives
  - ✓Filters
  - ✓Services, Factories

- Często wykorzystywane usługi
  - ✓$scope - obserwatory, obsługa zdarzeń
  - ✓$http - zapytania HTTP
  - ✓$q - obsługa Promise
  - ✓$timeout, $location, $window
- AngularJS w szablonach HTML
  - ✓data bindings
  - ✓aktualizowanie danych z ng-model
  - ✓operowanie na kolekcjach danych
  - ✓ukrywanie elementów na interfejsie
  - ✓obsługa zadarzeń
- Przydatne narzędzia podczas developmentu

Organizacja struktur projektu

Struktura plikowa projektu

`package.json` – pakiety środowiska dev.

`bower.json` – pakiety wykorzystywane w UI

`Gruntfile.js` – automatyzacja procesów

`npm-shrinkwrap.json` – locking wersji libów

`fake_*` - konfiguracja fake'owego API

`README.md` – podstawowa dokumentacja

`app/` - katalog z kodem aplikacji

`tests/` - katalog z testami automatycznymi

Struktura kodu aplikacji

`index.html` – inkludowanie plików aplikacji

`styles/` - css aplikacji

`views/` - szablony HTML

`scripts/app.js` – konfiguracja aplikacji

`scripts/*/` - katalogi z kodem aplikacji

`test/spec/*/` - katalogi z testami aplikacji

We make IT JCommerce

Elementy AngularJS

Routing

Konfiguracja routingu modułu AngularJS jest dokonywana za pomocą `$routeProvider`. Dostępne jest m. in.

- Wiązanie kontrolerów z szablonami
- Przekierowania
- Definiowanie resolve'erów
- Definiowanie `$routeParams`

```
$routeProvider
  .when('/kittens', {
    templateUrl: 'views/kittens/index.html',
    controller: 'KittensIndexCtrl',
    controllerAs: 'vm'
  })
  .when('/kittens/:id', {
    templateUrl: 'views/kittens/show.html',
    controller: 'KittensShowCtrl',
    controllerAs: 'vm'
  })
  .otherwise({
    redirectTo: '/kittens'
  });
```

Controllers

Odpowiadają za pobieranie i przetwarzanie danych poprzez usługi oraz udostępniają dane w szablonach

```
angular.module('kittensApp')
  .controller('KittensIndexCtrl', function (Kitten) {
    var vm = this;
    vm.collection = [];

    activate();

    function activate() {
      Kitten.getList().then(assignCollection);
    }

    function assignCollection(collection) {
      vm.collection = collection;
    }
  });
```

Directives

Dyrektywy to najczęściej komponenty, które odpowiadają charakterystycznym elementom DOM lub atrybutom. Mogą posiadać:

- Własny kontroler
- Własny szablon
- Własny scope

```
angular.module('kittensApp')
  .directive('kittenRow', function () {
    return {
      restrict: 'E',
      templateUrl: 'views/kittens/row.html',
      scope: {resource: '='}
    };
  });
```

Filters

Filtry to funkcje, które są najczęściej wykorzystywane bezpośrednio w szablonach HTML do modyfikacji wyświetlanych danych.

```
angular.module('kittensApp')
  .filter('onOff', function(input){
       return input ? 'On' : 'Off';
  });
```

```
<div>
  <label>Notifications</label>
  <p>{{notifications_enabled | onOff}}</p>
</div>
```

Services

Services / Factories to usługi, które mogą być wstrzyknięte do kontrolerów / dyrektyw. W usługach powinniśmy zawierać logikę aplikacji, tak by kod w kontrolerach / dyrektywach był minimalistyczny.

```
angular.module('kittensApp')
  .factory('openUrlService',
    function ($timeout, $window) {
    var url = '';

    function goTo(newUrl) {
      url = newUrl;
      _displayWarning();
      $timeout(_redirect, 5000);
    }

    function _displayWarning() {
      alert(url + ' will be opened in 5 seconds');
    }

    function _redirect() {
      $window.open(url);
    }

    return {goto: goTo};
  });
```

Services

Services / Factories to usługi, które mogą być wstrzyknięte do kontrolerów / dyrektyw. W usługach powinniśmy zawierać logikę aplikacji, tak by kod w kontrolerach / dyrektywach był minimalistyczny.

```
angular.module('kittensApp')
  .service('openUrlService',
    function ($timeout, $window) {
    this.url = '';

    this.goTo = function(newUrl) {
      this.url = newUrl;
      this._displayWarning();
      $timeout(this._redirect.bind(this), 5000);
    }

    this._displayWarning = function() {
      alert(url + ' will be opened in 5 seconds');
    }

    this._redirect = function() {
      $window.open(this.url);
    }
  });
```

Często wykorzystywane usługi

$scope

$scope reprezentuje kontekst dla danego kontrolera / dyrektywy. Przypisane do niego wartości są dostępne w szablonach HTML. Poprzez $scope możemy realizować zdarzenia w aplikacji oraz korzystać z obserwatorów.

- $emit – emitowanie eventów
- $broadcast – broadcast eventów
- $on – nasłuchiwanie eventów
- $watch – obserwowanie obiektów
- $watchCollection – obserwowanie kolekcji
- $digest – rozpoczęcie cyklu przeliczania data bindings.

```
angular.module('kittensApp')
  .controller('ResourceCtrl',
    function ($scope) {
    var vm = this;

    function save() {
      vm.resource.save().then(onSaved);
    }

    function onSaved() {
      $scope.$emit('resource-saved', vm.resource)
    }
  });
```

$scope

$scope reprezentuje kontekst dla danego kontrolera / dyrektywy. Przypisane do niego wartości są dostępne w szablonach HTML. Poprzez $scope możemy realizować zdarzenia w aplikacji oraz korzystać z obserwatorów.

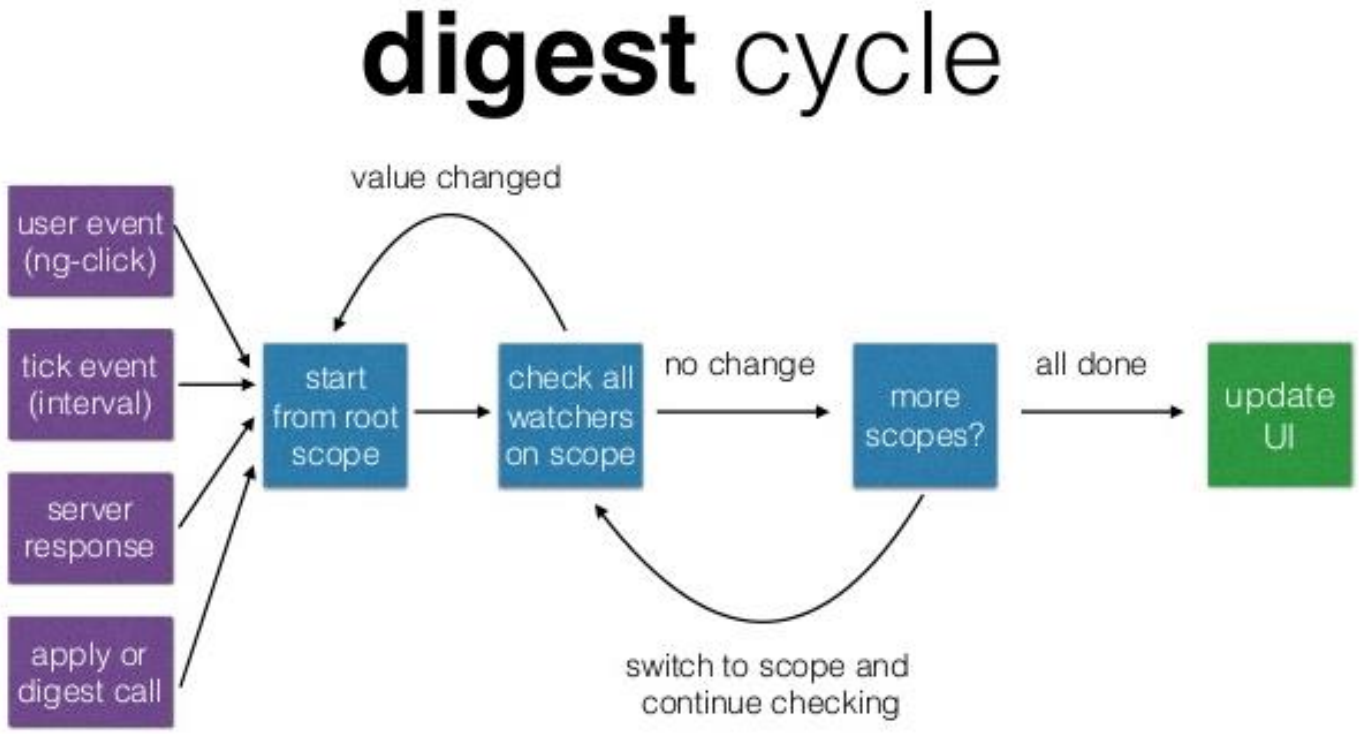
- $emit – emitowanie eventów
- $broadcast – broadcast eventów
- $on – nasłuchiwanie eventów
- $watch – obserwowanie obiektów
- $watchCollection – obserwowanie kolekcji

```
angular.module('kittensApp')
  .controller('CollectionCtrl',
    function ($scope) {
    var vm = this;
    vm.collection = [];

    function addToList(resource) {
      vm.collection.push(resource);
    }

    $scope.$on('resource-saved', addToList);
  });
```

$scope

Przeliczanie data bindings (cykl $digest) w scopie następuje w przypadku:

- zdarzenia na UI
- response'a zapytania HTTP
- zmiana wartości ngModel
- wywołanie $timeout
- wywołanie $scope.$digest() lub $scope.$apply()



$http

Usługa pozwalająca na realizację zapytań HTTP np. do API aplikacji.

```
angular.module('kittensApp')
  .controller('CollectionCtrl', function ($http) {
    var vm = this;
    vm.collection = [];

    activate();

    function activate() {
     $http.get('/api/kittens').then(_assignToCollection);
    }

    function _assignToCollection(collection) {
      vm.collection = collection;
    }
  });
```

$q

Obsługa promises, która udostępnia metody:

- resolve
- reject
- all

```
angular.module('kittensApp')
  .controller('CollectionCtrl', function ($q) {
    var vm = this;
    vm.collection = [];
    vm.saveAll = saveAll;

    function saveAll() {
      var promises = vm.collection.map(function(resource) {
        return resource.save();
      });
      $q.all(promises).then(_onSuccess, _onError);
    }

    function _onSuccess() {
      alert('All resources are saved!');
    }

    function _onError() {
      alert('Something went wrong!');
    }
  });
```

$timeout
$window
$location

Wrappery dla funkcji / obiektów dostępnych w window.

```
angular.module('kittensApp')
  .controller('CollectionCtrl', function ($q) {
    var vm = this;
    vm.resource = {};
    vm.save = save;

    function save() {
      vm.resource.save().then(_openResource);
    }

    function _openResource() {
      $location.path('/kittens/'+vm.resource.id);
    }
  });
```

AngularJS w szablonach HTML

Data bindings

Wyświetlanie danych w szablonach jest możliwe poprzez użycie podwójnego nawiasu wąsatego. Dodatkowo istnieje kilka dyrektywch takich jak ngSrc, ngHref, ngClass.

Podwójny dwukropek wskazuje, że wstawione dane nie powinny być przeliczane.

```
<div>
  <h4>{{::resource.title}}</h4>
  <p>Submitted by {{resource.submitted_by}}</p>
  <div>
      <p>{{resource.commentsLabel()}}</p>
      <img ng-src="{{resource.url}}" />
  </div>
</div>
```

ngModel

ngModel pozawala na powiązanie atrybutu obiektu z kontrolką edycyjną na interfejsie.

```
<form ng-submit="vm.save()">
  <div>
    <label>Title</label>
    <input type="text" ng-model="vm.resource.title">
  </div>
  <button type="submit">Add</button>
</form>
```

ngRepeat, ngShow, ngHide, ngIf

ngRepeat pozwala na renderowanie kolekcji danych.

```
<div ng-repeat="resource in vm.collection">
  <kitten-row resource="resource"></kitten-row>
</div>
```

ngShow, ngHide określa czy node powinien być urkyty.

```
<button ng-click="shownContact = !shownContact"></button>
<p ng-show="shownContact">{{resource.phone}}</p>
```

ngIf określa czy node powinien znajdować się w DOMie.

```
<button ng-click="shownContact = !shownContact"></button>
<p ng-if="shownContact">{{resource.phone}}</p>
```

Obsługa zdarzeń

Za pomocą atrybutów możemy obsługę zdarzeń. Często wykorzystywanym zdarzenem jest ng-click.

```
<button type="button" ng-click="vm.like()">
  I like it!
</button>
```

```
angular.module('kittensApp')
  .controller('KittensRowCtrl', function () {
    var vm = this;
    vm.like = like;

    function like() {
      vm.resource.like().then(onSuccess, onFailure);
    }

    function onFailure() { alert('Uh oh!'); }
    function onSuccess() { alert('Liked!'); }
  });
```

Przydatne narzędzia do developmentu

Narzędzia do developmentu

- Chrome developers console, debugger
- Chrome extensions
  - AngularJS watchers
  - AngularJS Batarang
- jshint
- karma-coverage

Pytania

We make IT JCommerce

We make IT JCommerce

I ty możesz zostać niezależnym developerem!

Tomasz Borowski

FrontEnd Developer

tomasz.borowski@jcommerce.pl