**Importando o ngResource**

Quando estamos utilizando endpoints REST no nosso back-end, o $http acaba não conseguindo esconder a complexidade deste tipo de serviço. Para este tipo de padrão, o Angular nos oferece o módulo ngResource. Sobre este módulo, julgue as seguintes afirmativas :

1) Ele **não** precisa ser importado no módulo principal da aplicação, pois já estamos utilizando o parâmetro $resource no controller que o utiliza, como abaixo:

// public/js/controllers/fotos-controller.js  
angular.module('alurapic').controller('FotosController', function($scope, $resource) {  
 // código omitido   
});

2) Para podermos utilizar o módulo ngResource, devemos importar o script dele na nossa view, como abaixo:

<!-- public/index.html -->  
<!DOCTYPE html>  
<html lang="pt-br" ng-app="alurapic">  
 <head>  
 <script src="js/lib/angular.min.js"></script>  
 <script src="js/lib/angular-resource.min.js"></script>  
 <script src="js/main.js"></script>  
 </head>  
 <body>  
 // código omitido  
 </body>  
</html>

3) Deve-se importar o módulo no nosso main.js antes de utilizá-lo.

// public/js/main.js  
angular.module('alurapic', ['minhasDiretivas', 'ngAnimate', 'ngRoute', 'ngResource'])  
 .config(function($routeProvider, $locationProvider) {  
 // código omitido   
 });

4) Não é necessário importar o script do ngResource, pois o mesmo não existe, já que este módulo faz parte do core do Angular e já está incluído no angular.min.js.

* Apenas 3 e 4 são verdadeiras.
* 1 e 3 são falsas.
* 2 e 3 são falsas.
* **2 e 3 são verdadeiras.**

**Explicação**

**O primeiro passo para usarmos ngResource é importar o script do módulo ngResoure em nossa viewindex.html.**

**<!-- public/index.html -->  
<!DOCTYPE html>  
<html lang="pt-br" ng-app="alurapic">  
 <head>  
 <script src="js/lib/angular.min.js"></script>  
 <script src="js/lib/angular-resource.min.js"></script>  
 <script src="js/main.js"></script>  
 </head>  
 <body>  
 // código omitido  
 </body>  
</html>**

**O segundo, não menos importante, é declararmos ngResource como dependência do módulo principal da aplicação:**

**// public/js/main.js  
  
angular.module('alurapic', ['minhasDiretivas', 'ngAnimate', 'ngRoute', 'ngResource'])  
 .config(function($routeProvider, $locationProvider) {  
  
 // código omitido   
 });**

**E por último, podemos começar a modificar o nosso FotosController para utilizar o $resource no local do$http:**

**// public/js/controllers/fotos-controller.js  
  
angular.module('alurapic').controller('FotosController', function($scope, $resource) {  
 var recursoFoto = $resource('/v1/fotos/:fotoId');  
  
 $scope.fotos = [];  
 $scope.filtro = '';  
 $scope.mensagem = '';  
  
 // código posterior omitido   
});**

**Criando um serviço!**

**Quando precisamos utilizar um mesmo recurso em diversos controllers diferentes, o Angular nos dá a opção de criar um serviço, através da função factory, que contenha este recurso, disponibilizando-o para os controllers através de injeção de dependência.**

**Considerando o seguinte recurso:**

**$resource('/v1/fotos/:fotoId', null, {  
 'update' : {  
 method: 'PUT'  
 }  
 });**

**Se queremos disponibilizá-lo como um serviço chamado recursoFoto para os nossos controllers, qual dos códigos abaixo poderemos utilizar para criar este serviço?**

1. **// public/js/servicos/meus-servicos.js  
   angular.module('meusServicos', ['ngResource'])  
    .factory(function($resource) {  
     
    $resource('/v1/fotos/:fotoId', null, {  
    'update' : {   
    method: 'PUT'  
    }  
    });  
    });**
2. **// public/js/servicos/meus-servicos.js  
   angular.module('meusServicos', ['ngResource'])  
    .factory('recursoFoto', $resource {  
     
    return $resource('/v1/fotos/:fotoId', null, {  
    'update' : {   
    method: 'PUT'  
    }  
    });  
    });**
3. **// public/js/servicos/meus-servicos.js  
   angular.module('meusServicos',['ngResource'])  
    .factory('recursoFoto', function() {  
     
    return $resource('/v1/fotos/:fotoId', null, {  
    'update' : {   
    method: 'PUT'  
    }  
    });  
    });**
4. **// public/js/servicos/meus-servicos.js  
   angular.module('meusServicos', ['ngResource'])  
    .factory('recursoFoto', function($resource) {  
     
    return $resource('/v1/fotos/:fotoId', null, {  
    'update' : {   
    method: 'PUT'  
    }  
    });  
    });**

**Resposta: d**

**Explicação**

**// public/js/servicos/meus-servicos.js  
angular.module('meusServicos', ['ngResource'])  
 .factory('recursoFoto', function($resource) {  
 return $resource('/v1/fotos/:fotoId', null, {  
 'update' : {   
 method: 'PUT'  
 }  
 });  
 });**

**Utilizamos a função factory para criar um serviço, ela recebe como parâmetro o nome do serviço e a uma função que deve retornar um objeto.**

**Aprendendo o $resource**

**Sobre o $resource, julgue as seguintes afirmativas:**

**1) O $resource suporta todos os verbos HTTP que $http suporta.**

**2) A seguinte chamada da funçao query está correta:**

**var recursoFoto = $resource('/v1/fotos/:fotoId');**

**recursoFoto.query(success(fotos) {  
 $scope.fotos = fotos;  
}, error(error(erro)) {  
 console.log(erro);  
});**

**3) Uma das grandes vantagens de se utilizar o $resource é que ele provê um alto nível de abstração em relação ao seu similar $http, facilitando quando estamos trabalhando com um back-end RESTful.**

* Todas as afirmativas são falsas.
* Apenas 1 é falsa.
* 1 e 3 são falsas.
* **Apenas 3 é verdadeira.**

**Explicação**

**Apenas a terceira afirmativa é verdadeira.**

**O $resource não suporta o verbo HTTP PUT, porém é possível implementá-lo. Por fim, a declaração correta da função query é:**

**recursoFoto.query(function(fotos) {  
 $scope.fotos = fotos;  
}, function(erro) {  
 console.log(erro);  
});**