Question 1. Answers highlighted in red.

<body data-ng-app="app">

<div ng-controller=“TitleController">

<button id="get-items-button" ng-click="getTitles()">Get Pets</button>

<p>Look at the list of titles!</p>

<!--this table shows the titles we get from our service-->

<table cellpadding="0" cellspacing="0">

<thead> <tr> <th>Code</th> <th>Title</th> </tr> </thead>

<tbody>

<tr **ng-repeat=”title in titles”**>

<td>{{title.titleCode}}</td>

<td>{{title.title}}</td>

</tr>

</tbody>

</table>

</div>

<script>

(function () {

//create the module

angular.module('app', [])

//add controller

.controller(‘TitleController', function ((**$scope, $http**) {

//declare an array of items. this will get populated with our ajax call

$scope.titles = [];

//declare an action for our button

$scope. getTitles = function () {

var url = “/api/Codes/Titles/”;

//perform ajax call and on success copy the data we get into the items array so that angular can track the object and bind it automatically. On error alert the message “Error getting data”.

**$http.get(url)  
    .then(function(response) {   
        $scope.titles = response.data;  
    }, function(response) {  
        alert(“Error getting data”)  
    });**

} });

})();

</script>

Question 2:

1. karma, karma-chrome-launcher, karma-cli, karma-jasmine, jasmine, jasmine-core
2. Collection of method like a class.
3. What is great about Angularjs way of injecting dependencies is that it only creates an instance of a service once. Then the next time you call the service, you will get the same object unless you clear the cache. It also minimize hard coding of services on your scripts which makes your scripts cleaner and easier to maintain.
4. Create table-directive.html template

<table cellpadding="0" cellspacing="0">

<thead> <tr> <th>Code</th> <th>Title</th> </tr> </thead>

<tbody>

<tr ng-repeat=”item in data”>

<td>{{ item.titleCode}}</td>

<td>{{ item.title}}</td>

</tr>

</tbody>

</table>

On script we will use the html template:

var tableDirective = function() {

return {

scope: {

data: ‘=’

},

templateUrl: ‘table-directive.html’

}

};

angular.module(‘app’).directive(‘tableDirective’, tableDirective);

then passing $scope.titles on the directive

<table-directive “data=titles”></table-directive>