AngularJS is an open source web application framework. It was originally developed in 2009 by Misko Hevery and Adam Abrons. It is now maintained by Google. Its latest version is 1.4.3.

Official Definition:

AngularJS is a structural framework for dynamic web apps. It lets you use HTML as your template language and lets you extend HTML's syntax to express your application's components clearly and succinctly. Angular's data binding and dependency injection eliminate much of the code you currently have to write. And it all happens within the browser, making it an ideal partner with any server technology.

Why AngularJS is more popular now a days:

AngularJS data binding and dependency injection eliminate much of the code you currently have to write. And it all happens within the browser, making it an ideal partner with any server technology.

Overall AngularJS is a Frame Work to build large scale and high performance web apps while keeping them as easy to maintain.

AngularJS applications can run on all major browsers and smart phones including Android and IOS based phones or tablets.

We have so many advantages of AngularJS like data binding, less code, re usable components, dependency injection etc. we have few drawbacks as well.

Dis Advantages:

Not Secure: Being JavaScript Only Frame Work application written in AngularJS are not safe. Server side authentication/ authorization are must to keep an application secure.

Not Degradable: If your application user disables Java Script then user will just see the basic page and nothing more.

AngularJS Components:

The AngularJS framework can be divided into following three major parts.

. ng-app : This directive defines and links an AngularJS application to HTML.

. ng-model: This directive binds the values of AngularJS application data to HTML input controls.

. ng-bind : This directive binds the AngularJS Application data to HTML tags.

Example Code Walk Through:

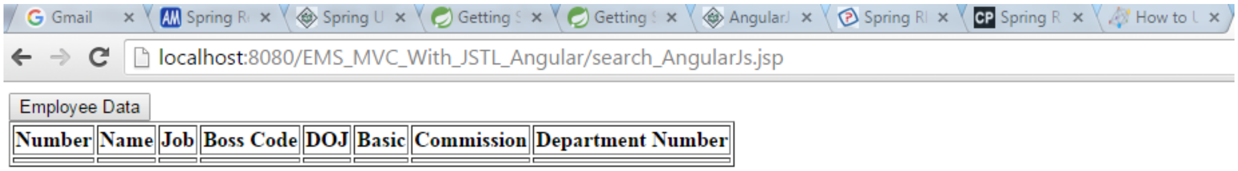
```
<!DOCTYPE html>
<html><head><meta charset="ISO-8859-1">
<title>AJAX with Servlets using AngularJS</title>
<script type="text/javascript" src="js/angular.min.js"></script>
<script>
	var app = angular.module('myApp', []);
	function MyController($scope, $http) {
		$scope.getDataFromServer = function() {
			$http({
				method : 'POST',
				url : 'SearchController1'
			}).success(function(data, status, headers, config) {
				$scope.employeeVO = data;
			}).error(function(data, status, headers, config) {
				// called asynchronously if an error occurs
			// or server returns response with an error status.
			});
		};
	};
</script></head>
<body><div data-ng-app="myApp">
		<div data-ng-controller="MyController">
<button data-ng-click="getDataFromServer()">Employee Data</button>
			<table border="1">
			<tr><th>Number</th>
			<th>Name</th>
			<th>Job</th>
			<th>Boss Code</th>
			<th>DOJ</th>
			<th>Basic</th>
			<th>Commission</th>
			<th>Department Number</th>
			</tr> <tr><td>{{employeeVO.empNumber}}</td>
			<td>{{employeeVO.empName}}</td>
			<td>{{employeeVO.empJob}}</td>
			<td>{{employeeVO.empBossCode}}</td>
			<td>{{employeeVO.empDoj}}</td>
			<td>{{employeeVO.empSalary}}</td>
			<td>{{employeeVO.empComm}}</td>
			<td>{{employeeVO.empDeptNumber}}</td>
			</tr>
		</table>
	</div></div></body>
</html>
```

Execution:

Save the above code as *myfirstexample.html* and open it in any browser. You will see an output as below –

When the page is loaded in the browser, following things happen –

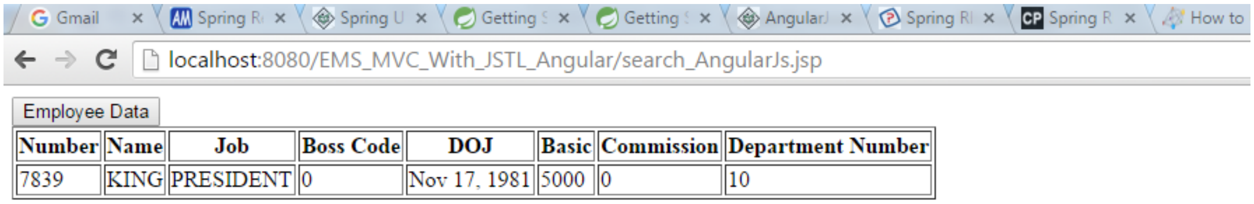
- HTML document is loaded into the browser, and evaluated by the browser. AngularJS JavaScript file is loaded, the angular *global* object is created. Next, JavaScript which registers controller functions is executed.
- Next AngularJS scans through the HTML to look for AngularJS apps and views. Once view is located, it connects that view to the corresponding controller function.
- Next, AngularJS executes the controller functions. It then renders the views with data from the model populated by the controller. The page is now ready.
- Normally sending data and receiving the data to server and from the server usually happens in JSON Format.
- JSON means Java Script Object Notation and it is a combination of name and value pairs.



localhost:8080/EMS_MVC_With_JSTL_Angular/search_AngularJs.jsp

Employee Data

Number	Name	Job	Boss Code	DOJ	Basic	Commission	Department Number



localhost:8080/EMS_MVC_With_JSTL_Angular/search_AngularJs.jsp

Employee Data

Number	Name	Job	Boss Code	DOJ	Basic	Commission	Department Number
7839	KING	PRESIDENT	0	Nov 17, 1981	5000	0	10

Deep Linking: Deep Linking allows you to encode the state of application in the URL so that it can be book marked. The application can then be restored from the URL to the same state.

Servlet Code:

```
   SearchBO searchBO = new SearchBO();

EmployeeVO employeeVO = searchBO.searchEmployee(empNumber);
                String json = new Gson().toJson(employeeVO);
                response.setContentType("application/json");
                response.getWriter().write(json);
```

Example Code Walk Through:

Servlet Code:

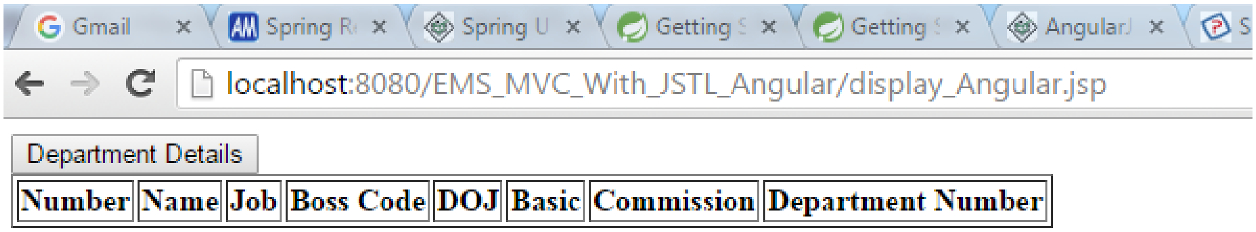
DisplayBO displayBO=new DisplayBO();

```
List<EmployeeVO>
empList=displayBO.displayEmployeesList(empDeptNumber);
                String json=new Gson().toJson(empList);
                response.setContentType("application/json");
                response.getWriter().write(json);
```

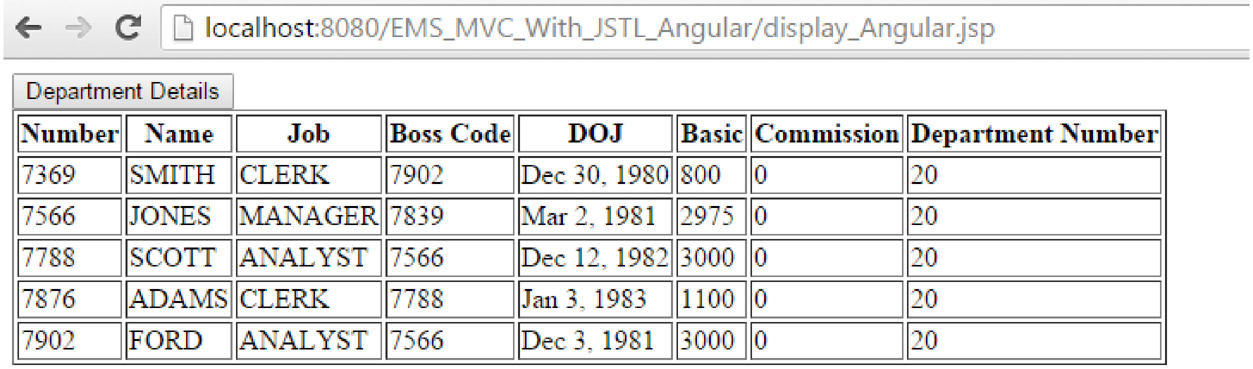
Html Page:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>AJAX with Servlets using AngularJS</title>
<script type="text/javascript" src="js/angular.min.js"></script>
<script>
     var app = angular.module('myApp', []);
     function MyController($scope, $http) {
          $scope.getDataFromServer = function() {
                $http({
                     method : 'POST',
                     url : 'DisplayController1'
                }).success(function(data, status, headers, config) {
                     $scope.empList=data;
                }).error(function(data, status, headers, config) {
                     // called asynchronously if an error occurs
                // or server returns response with an error status.
                });
          };
     };
</script></head>
```

```
<body>
     <div data-ng-app="myApp">
          <div data-ng-controller="MyController"><button data-ng-click="getDataFromServer()">Department Details</button>
                <table border="1">
                <tr><th>Number</th>
                <th>Name</th>
                <th>Job</th>
                <th>Boss Code</th>
                <th>DOJ</th>
                <th>Basic</th>
                <th>Commission</th>
                <th>Department Number</th>
                </tr>
            <tr ng-repeat = "employeeVO in empList">
                <td>{{employeeVO.empNumber}}</td>
                <td>{{employeeVO.empName}}</td>
                <td>{{employeeVO.empJob}}</td>
                <td>{{employeeVO.empBossCode}}</td>
                <td>{{employeeVO.empDoj}}</td>
                <td>{{employeeVO.empSalary}}</td>
                <td>{{employeeVO.empComm}}</td>
                <td>{{employeeVO.empDeptNumber}}</td>
                </tr>
          </table></div></div>
</body>
</html>
```



Number	Name	Job	Boss Code	DOJ	Basic	Commission	Department Number


localhost:8080/EMS_MVC_With_JSTL_Angular/display_Angular.jsp
Department Details

Number	Name	Job	Boss Code	DOJ	Basic	Commission	Department Number
7369	SMITH	CLERK	7902	Dec 30, 1980	800	0	20
7566	JONES	MANAGER	7839	Mar 2, 1981	2975	0	20
7788	SCOTT	ANALYST	7566	Dec 12, 1982	3000	0	20
7876	ADAMS	CLERK	7788	Jan 3, 1983	1100	0	20
7902	FORD	ANALYST	7566	Dec 3, 1981	3000	0	20

AngularJS directives:

AngularJS directives are used to extend HTML. These are special attributes starting with ng- prefix. We're going to discuss following directives –

- **ng-app** – This directive starts an AngularJS Application.
- **ng-init** – This directive initializes application data.
- **ng-model** – This directive defines the model that is variable to be used in AngularJS.
- **ng-repeat** – This directive repeats html elements for each item in a collection.

AngularJS Expressions:

Expressions are used to bind application data to html. Expressions are written inside double braces like {{ expression}}. Expressions behave in same way as ng-bind directives. An AngularJS application expression is pure javascript expressions and outputs the data where they are used.

```
Ex: <td>{{employeeVO.empNumber}}</td>
```

AngularJS Controller:

AngularJS application mainly relies on controllers to control the flow of data in the application. A controller is defined using ng-controller directive.

A controller is a JavaScript object containing attributes/properties and functions. Each controller accepts $scope as a parameter which refers to the application/module that controller is to control.

Note: We can also defined the controller object in separate js file and refer that file in the html page.

```
Ex: <div data-ng-controller="MyController">
```

AngularJS Filters:

Filters are used to change modify the data and can be clubbed in expression or directives using pipe character. Following is the list of commonly used filters.

Sr.No.	Name	Description
1	uppercase	converts a text to upper case text.
2	lowercase	converts a text to lower case text.
3	currency	formats text in a currency format.
4	filter	filter the array to a subset of it based on provided criteria.
5	orderby	orders the array based on provided criteria.

Tables:

Table data is normally repeatable by nature; ng-repeat directive can be used to draw the table easily.

```
<tr ng-repeat = "employeeVO in empList">
           <td>{{employeeVO.empNumber}}</td>
           <td>{{employeeVO.empName}}</td>
           <td>{{employeeVO.empJob}}</td>
           <td>{{employeeVO.empBossCode}}</td>
           <td>{{employeeVO.empDoj}}</td>
           <td>{{employeeVO.empSalary}}</td>
           <td>{{employeeVO.empComm}}</td>
           <td>{{employeeVO.empDeptNumber}}</td>
</tr>
```

HTML DOM Elements:

1) ng-disabled 2) ng-show 3) ng-hide 4) ng-click

Example:

```
<button data-ng-click="getDataFromServer ()">

   Department Details

  </button>
```

AngularJS Modules:

AngularJS supports modular approach. Modules are used to separate logics say services, controllers, application etc. and keep the code clean. We define modules in separate js files and name them as per the module.js file. In this example we're going to create two modules.

- **Application Module** – used to initialize an application with controller(s).

  ```
  var app = angular.module('myApp', []);
  ```

- **Controller Module** – used to define the controller.

  ```
  var app = angular.module('myApp', []);
  function MyController($scope, $http) {
        $scope.getDataFromServer = function() {
              $http({
                    method : 'POST',
                    url : 'DisplayController1'
              }).success(function(data, status, headers, config) {
                    $scope.empList=data;
              }).error(function(data, status, headers, config) {
                    // called asynchronously if an error occurs
              // or server returns response with an error status.
              });
        };
  };
  ```

Use Like This:

```
<div ng-app = "mainApp" ng-controller = "studentController">
  ...
  <script src = "mainApp.js"></script>
  <script src = "studentController.js"></script>


</div>
```

AngularJS – Forms:

AngularJS enriches form filling and validation. We can use ng-click to handle AngularJS click on button and use $dirty and $invalid flags to do the validations in seamless way.

Use novalidate with a form declaration to disable any browser specific validation. Forms controls makes heavy use of Angular events.

Validate Data:

Following can be used to track error.

- **$dirty** − states that value has been changed.
- **$invalid** − states that value entered is invalid.
- **$error** − states the exact error.

Example Program:

FormValidationExample.html:

```
<html>
  <head>
    <title>Angular JS Forms</title>
<script src =
"http://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js">
</script>
    <style>
      table, th , td {
        border: 1px solid grey;
        border-collapse: collapse;
        padding: 5px;
      }
      table tr:nth-child(odd) {
        background-color: #f2f2f2;
      }
      table tr:nth-child(even) {
        background-color: #ffffff;
      }
    </style>

  </head>
```

```
 <body>
      <h2>AngularJS Sample Application</h2>
      <div ng-app = "mainApp" ng-controller = "studentController">
 <form name = "studentForm" novalidate>
 <table border = "0">
 <tr><td>Enter first name:</td>
<td><input name = "firstname" type = "text" ng-model = "firstName"
required>
<span style = "color:red" ng-show = "studentForm.firstname.$dirty &&
studentForm.firstname.$invalid">
<span ng-show = "studentForm.firstname.$error.required">First Name is
required.</span></span></td></tr>
<tr><td>Enter last name: </td>
<td><input name = "lastname"  type = "text" ng-model = "lastName"
required>
<span style = "color:red" ng-show = "studentForm.lastname.$dirty &&
studentForm.lastname.$invalid">
<span ng-show = "studentForm.lastname.$error.required">Last Name is
required.</span></span></td></tr>
<tr><td>Email: </td>
<td><input name = "email" type = "email" ng-model = "email" length =
"100" required>
<span style = "color:red" ng-show = "studentForm.email.$dirty &&
studentForm.email.$invalid">
<span ng-show = "studentForm.email.$error.required">Email is
required.</span>
<span ng-show = "studentForm.email.$error.email">Invalid email
address.</span></span></td></tr><tr> <td>
<button ng-click = "reset()">Reset</button></td><td>
<button ng-disabled = "studentForm.firstname.$dirty &&
 studentForm.firstname.$invalid || studentForm.lastname.$dirty &&
 studentForm.lastname.$invalid || studentForm.email.$dirty &&
 studentForm.email.$invalid" ng-click="submit()">Submit</button>
 </td></tr></table></form></div>
 <script>
         var mainApp = angular.module("mainApp", []);
         mainApp.controller('studentController', function($scope) {
            $scope.reset = function(){
               $scope.firstName = "Srinivasa Reddy";
               $scope.lastName = " Challa";
               $scope.email = "urtrainer.java@gmail.com";
            }
            $scope.reset();
         });
      </script></body></html>
```

AngularJS – Includes:

HTML does not support embedding html pages within html page. To achieve this functionality following ways are used –

- **Using Ajax** – Make a server call to get the corresponding html page and set it in innerHTML of html control.
- **Using Server Side Includes** – JSP, PHP and other web side server technologies can include html pages within a dynamic page.

Using AngularJS, we can embedded HTML pages within a HTML page using ng-include directive.

```
<div ng-app = "" ng-controller = "studentController">
  <div ng-include = "'main.htm'"></div>
  <div ng-include = "'subjects.htm'"></div>
</div>
```

AngularJS – Ajax :

AngularJS provides $http control which works as a service to read data from the server. The server makes a database call to get the desired records. AngularJS needs data in JSON format. Once the data is ready, $http can be used to get the data from server in the following manner –

```
function studentController($scope,$http) {
var url = "data.txt";

  $http.get(url).success( function(response) {
    $scope.students = response;
  });
}
```

Here, the file data.txt contains student records. $http service makes an Ajax call and sets response to its property students.*students* model can be used to draw tables in HTML.

Exmple Code:

data.txt:

```
[
   {
      "Name" : "Srinivasa Reddy",
      "RollNo" : 101,
      "Percentage" : "80%"
   },

   {
      "Name" : "Vasu ",
      "RollNo" : 201,
      "Percentage" : "70%"
   },

   {
      "Name" : "Challa",
      "RollNo" : 191,
      "Percentage" : "75%"
   },

   {
      "Name" : "Srinivas",
      "RollNo" : 111,
      "Percentage" : "77%"
   }
]
```

To execute this example, you need to deploy **Angular_JSON_AJAX.html** and ***data.txt*** file to a web server. Open the **Angular_JSON_AJAX.html** using the URL of your server in a web browser and see the result.

Angular_JSON_AJAX.html:

```
<html>
   <head>
      <title>Angular JS Includes</title>
      <style>
         table, th , td {
            border: 1px solid grey;
            border-collapse: collapse;
            padding: 5px;
         }
         table tr:nth-child(odd) {
            background-color: #f2f2f2;
         }
         table tr:nth-child(even) {
            background-color: #ffffff;
         }
      </style>
   </head>
   <body>
      <h2>AngularJS Sample Application</h2>
      <div ng-app = "" ng-controller = "studentController">
         <table> <tr>
               <th>Name</th>
               <th>Roll No</th>
               <th>Percentage</th>
            </tr>
            <tr ng-repeat = "student in students">
               <td>{{ student.Name }}</td>
               <td>{{ student.RollNo }}</td>
               <td>{{ student.Percentage }}</td>
            </tr> </table> </div>
      <script>
         function studentController($scope,$http) {
            var url = "data.txt";
            $http.get(url).success( function(response) {
               $scope.students = response;
            });
         }
      </script>
<script src =
"http://ajax.googleapis.com/ajax/libs/angularjs/1.2.15/angular.min.js">
</script>
   </body>
</html>
```

AngularJS - Views:

//TO DO

AngularJS - Scopes:

Scope is a special JavaScript object which plays the role of joining controller with the views. Scope contains the model data. In controllers, model data is accessed via $scope object.

```
<script>
  var mainApp = angular.module("mainApp", []);
  mainApp.controller("shapeController", function($scope) {
    $scope.message = "In shape controller";
    $scope.type = "Shape";
  });</script>
```

Following are the important points to be considered in above example.

- $scope is passed as first argument to controller during its constructor definition.
- $scope.message and $scope.type are the models which are to be used in the HTML page.
- We've set values to models which will be reflected in the application module whose controller is shapeController.
- We can define functions as well in $scope.

Scope Inheritance

Scope are controllers specific. If we define nested controllers then child controller will inherit the scope of its parent controller.

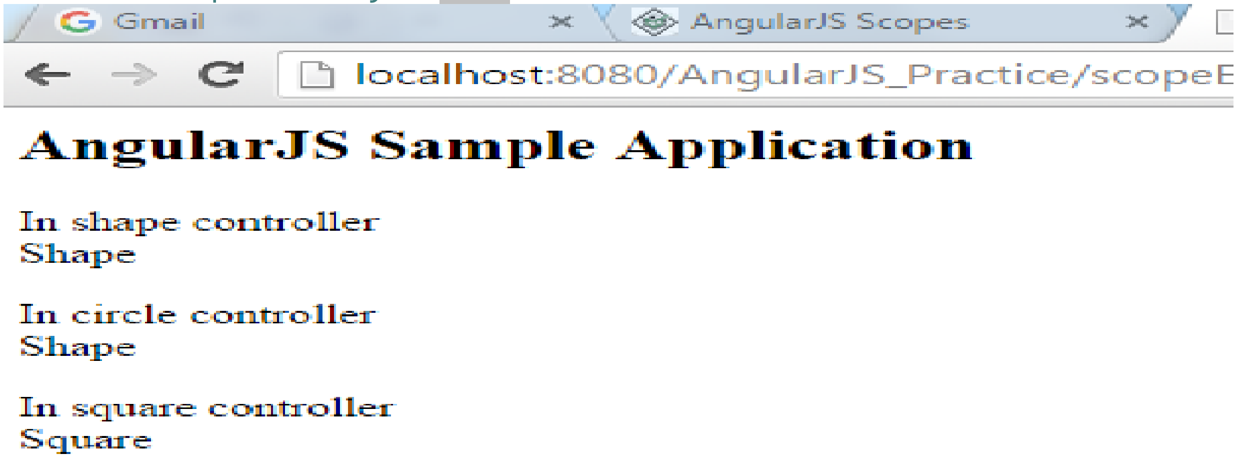
```
<script>
  var mainApp = angular.module("mainApp", []);
  mainApp.controller("shapeController", function($scope) {
    $scope.message = "In shape controller";
    $scope.type = "Shape";
  });
  mainApp.controller("circleController", function($scope) {
    $scope.message = "In circle controller";
  });    </script>
```

Following are the important points to be considered in above example.

- We've set values to models in shapeController.
- We've overridden message in child controller circleController. When "message" is used within module of controller circleController, the overridden message will be used.

Scope_Example.html:

```
<html>
   <head>
      <title>Angular JS Forms</title>
   </head>
   <body>
      <h2>AngularJS Sample Application</h2>
      <div ng-app = "mainApp" ng-controller = "shapeController">
         <p>{{message}} <br/> {{type}} </p>
         <div ng-controller = "circleController">
            <p>{{message}} <br/> {{type}} </p>
         </div>
         <div ng-controller = "squareController">
            <p>{{message}} <br/> {{type}} </p>
         </div>
      </div>
      <script src =
"http://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js">
</script> <script>
         var mainApp = angular.module("mainApp", []);
         mainApp.controller("shapeController", function($scope) {
            $scope.message = "In shape controller";
            $scope.type = "Shape";
         });
         mainApp.controller("circleController", function($scope) {
            $scope.message = "In circle controller";
         });
         mainApp.controller("squareController", function($scope) {
            $scope.message = "In square controller";
            $scope.type = "Square";
         });
      </script> </body></html>
```



AngularJS Sample Application

In shape controller
Shape

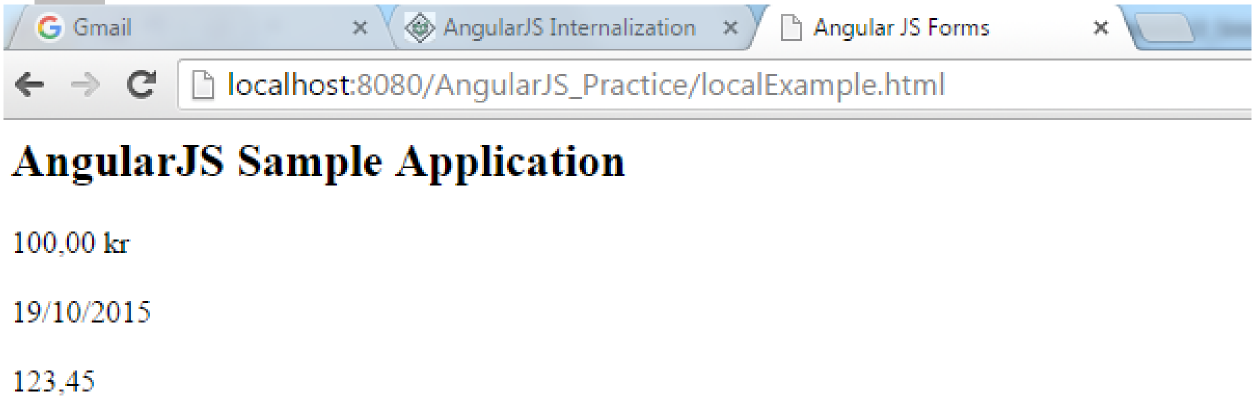
In circle controller
Shape

In square controller
Square

AngularJS – Internationalization:

AngularJS supports inbuilt internationalization for three types of filters currency, date and numbers. We only need to incorporate corresponding js according to locale of the country. By default it handles the locale of the browser. For example, to use Danish locale, use following script.

```
<html>
   <head>
      <title>Angular JS Forms</title>
   </head>
   <body>
      <h2>AngularJS Sample Application</h2>
      <div ng-app = "mainApp" ng-controller = "StudentController">
         {{fees | currency }}  <br/><br/>
         {{admissiondate | date }}  <br/><br/>
         {{rollno | number }}
      </div>
      <script src =
"http://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js">
</script>
     <script src = "https://code.angularjs.org/1.3.14/i18n/angular-
locale_da-dk.js"></script>
      <script>
         var mainApp = angular.module("mainApp", []);
         mainApp.controller('StudentController', function($scope) {
            $scope.fees = 100;
            $scope.admissiondate  = new Date();
            $scope.rollno = 123.45;
         });
      </script>
   </body>
</html>
```


Gmail
AngularJS Internalization
Angular JS Forms
localhost:8080/AngularJS_Practice/localExample.html

AngularJS Sample Application

100,00 kr

19/10/2015

123,45

AngularJS – Services:

AngularJS supports the concepts of "Separation of Concerns" using services architecture. Services are JavaScript functions and are responsible to do a specific task only. This makes them an individual entity which is maintainable and testable. Controllers, filters can call them as on requirement basis. Services are normally injected using dependency injection mechanism of AngularJS.

AngularJS provides many inbuilt services for example, $http, $route, $window, $location etc. Each service is responsible for a specific task for example, $http is used to make Ajax call to get the server data. $route is used to define the routing information and so on. Inbuilt services are always prefixed with $ symbol.

There are two ways to create a service.

- factory
- service

Using factory method

Using factory method, we first define a factory and then assign method to it.

```
var mainApp = angular.module("mainApp", []);
mainApp.factory('MathService', function() {
  var factory = {};
  factory.multiply = function(a, b) {
    return a * b
  }
  return factory;
});
```

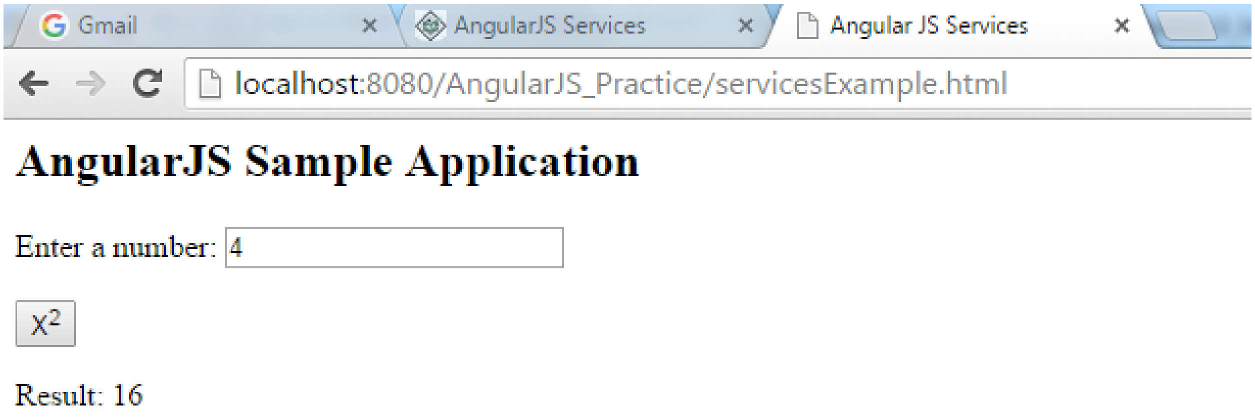
Using service method

Using service method, we define a service and then assign method to it. We've also injected an already available service to it.

```
mainApp.service('CalcService', function(MathService){
  this.square = function(a) {
    return MathService.multiply(a,a);
  }
});
```

servicesExample.html:

```
<html><head><title>Angular JS Services</title>
<script src =
"http://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js">
</script></head>
   <body><h2>AngularJS Sample Application</h2>
      <div ng-app = "mainApp" ng-controller = "CalcController">
  <p>Enter a number: <input type = "number" ng-model = "number" /></p>
         <button ng-click = "square()">X<sup>2</sup></button>
         <p>Result: {{result}}</p>
      </div><script>
         var mainApp = angular.module("mainApp", []);
         mainApp.factory('MathService', function() {
            var factory = {};
            factory.multiply = function(a, b) {
               return a * b
            }
            return factory;
         });
         mainApp.service('CalcService', function(MathService){
            this.square = function(a) {
               return MathService.multiply(a,a);
            }
         });
  mainApp.controller('CalcController', function($scope, CalcService) {
            $scope.square = function() {
               $scope.result = CalcService.square($scope.number);
            }
         });
      </script>
   </body>
</html>
```



AngularJS Sample Application

Enter a number: 4

X²

Result: 16

servicesExample2.html:

```
<html> <head><title>Angular JS Services</title>
 <script src =
"http://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js">
</script> </head>
   <body><h2>AngularJS Sample Application</h2>
      <div ng-app = "mainApp" ng-controller = "CalcController">
  <p>Enter a number: <input type = "number" ng-model = "number" /></p>
         <button ng-click = "square()">X<sup>2</sup></button>
         <p>Result: {{mulResult}}</p>
 <p>Enter a number: <input type = "number" ng-model = "number1" /></p>
 <p>Enter a number: <input type = "number" ng-model = "number2" /></p>
         <button ng-click = "add()">Add</button>
         <p>Result: {{addResult}}</p>
 </div><script>
         var mainApp = angular.module("mainApp", []);
         mainApp.factory('MathFactory', function() {
            var factory = {};
            factory.multiply = function(a, b) {
               return a * b
            }
            factory.add=function(a,b){
                return a+b;
            }
            return factory;
         });
         mainApp.service('CalcService', function(MathFactory){
            this.square = function(a) {
               return MathFactory.multiply(a,a);
            }
            this.add = function(a,b) {
                return MathFactory.add(a,b);
             }
         });
 mainApp.controller('CalcController', function($scope, CalcService) {
            $scope.square = function() {
               $scope.mulResult = CalcService.square($scope.number);
            }
            $scope.add = function() {
                $scope.addResult =
CalcService.add($scope.number1,$scope.number2);
             }
         });
      </script>
   </body></html>
```

AngularJS Dependency Injection:

Dependency Injection is a software design pattern in which components are given their dependencies instead of hard coding them within the component. This relieves a component from locating the dependency and makes dependencies configurable. This helps in making components reusable, maintainable and testable.

AngularJS provides a supreme Dependency Injection mechanism. It provides following core components which can be injected into each other as dependencies.

- value
- factory
- service
- provider
- constant

Value: value is simple JavaScript object and it is used to pass values to controller during config phase.

Factory: factory is a function which is used to return value. It creates value on demand whenever a service or controller requires. It normally uses a factory function to calculate and return the value.

Service: service is a singleton JavaScript object containing a set of functions to perform certain tasks. Services are defined using service () functions and then injected into controllers.

Provider: provider is used by AngularJS internally to create services, factory etc. during config phase (phase during which AngularJS bootstraps itself). Below mention script can be used to create MathService that we've created earlier. Provider is a special factory method with a method get () which is used to return the value/service/factory.

Constant: constants are used to pass values at config phase considering the fact that value cannot be used to be passed during config phase.

DependencyInjectionExample.html:

```
<html><head><title>AngularJS Dependency Injection</title></head>
   <body><h2>AngularJS Sample Application</h2>
      <div ng-app = "mainApp" ng-controller = "CalcController">
  <p>Enter a number: <input type = "number" ng-model = "number" /></p>
         <button ng-click = "square()">X<sup>2</sup></button>
         <p>Result: {{result}}</p>
      </div>
 <script src =
"http://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js">
</script>  <script>
         var mainApp = angular.module("mainApp", []);
         mainApp.config(function($provide) {
            $provide.provider('MathService', function() {
               this.$get = function() {
                  var factory = {};
                  factory.multiply = function(a, b) {
                     return a * b;
                  }
                  return factory;
               };
            });
         });    mainApp.value("defaultInput", 5);
                mainApp.factory('MathService', function() {
                var factory = {};
                factory.multiply = function(a, b) {
                return a * b;
                }
                return factory;
                });   mainApp.service('CalcService',
                      function(MathService){
                      this.square = function(a) {
                      return MathService.multiply(a,a);
                      }
                });   mainApp.controller('CalcController',
                      function($scope, CalcService, defaultInput) {
                      $scope.number = defaultInput;
                 $scope.result = CalcService.square($scope.number);
                 $scope.square = function() {
                 $scope.result = CalcService.square($scope.number);
                }
         });
      </script>
   </body>
</html>
```

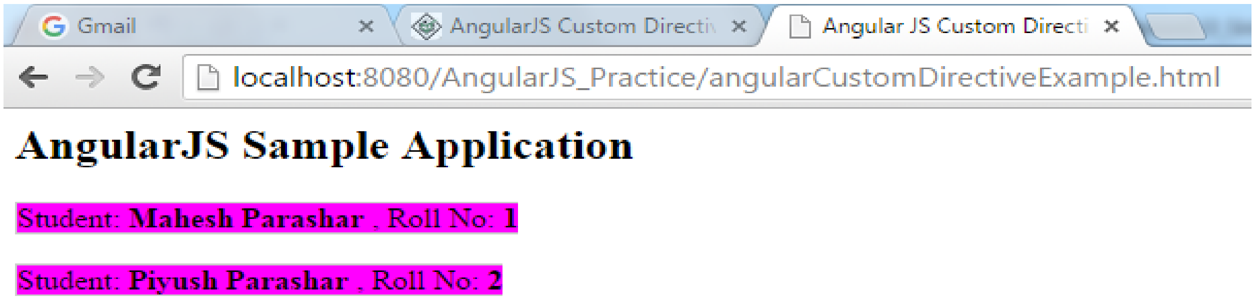
AngularJS - Custom Directives:

Custom directives are used in AngularJS to extend the functionality of HTML. Custom directives are defined using "directive" function. A custom directive simply replaces the element for which it is activated. AngularJS application during bootstrap finds the matching elements and do one time activity using its compile() method of the custom directive then process the element using link() method of the custom directive based on the scope of the directive. AngularJS provides support to create custom directives for following type of elements.

- **Element directives** – Directive activates when a matching element is encountered.
- **Attribute** – Directive activates when a matching attribute is encountered.
- **CSS** – Directive activates when a matching css style is encountered.
- **Comment** – Directive activates when a matching comment is encountered.

Define controller to update the scope for directive. Here we are using name attribute's value as scope's child.

```
mainApp.controller('StudentController', function($scope) {
   $scope.Mahesh = {};
   $scope.Mahesh.name = "Mahesh Parashar";
   $scope.Mahesh.rollno  = 1;
   $scope.Piyush = {};
   $scope.Piyush.name = "Piyush Parashar";
   $scope.Piyush.rollno  = 2;
});
```


Gmail
AngularJS Custom Directi
Angular JS Custom Directi
localhost:8080/AngularJS_Practice/angularCustomDirectiveExample.html

AngularJS Sample Application

Student: **Mahesh Parashar** , Roll No: **1**

Student: **Piyush Parashar** , Roll No: **2**

AngularJSCustomDirectiveExample.html:

```
<html>
   <head><title>Angular JS Custom Directives</title></head>
   <body><h2>AngularJS Sample Application</h2>
      <div ng-app = "mainApp" ng-controller = "StudentController">
         <student name = "Mahesh"></student><br/>
         <student name = "Piyush"></student>
      </div>
<script src =
"http://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js">
</script>
      <script>
         var mainApp = angular.module("mainApp", []);
         mainApp.directive('student', function() {
            var directive = {};
            directive.restrict = 'E';
            directive.template = "Student: <b>{{student.name}}</b> ,
Roll No: <b>{{student.rollno}}</b>";
            directive.scope = {
               student : "=name"
            }
            directive.compile = function(element, attributes) {
               element.css("border", "1px solid #cccccc");
     var linkFunction = function($scope, element, attributes) {
element.html("Student: <b>"+$scope.student.name +"</b> , Roll No:
<b>"+$scope.student.rollno+"</b><br/>");
                  element.css("background-color", "#ff00ff");
               }
               return linkFunction;
            }
            return directive;
         });
        mainApp.controller('StudentController', function($scope) {
            $scope.Mahesh = {};
            $scope.Mahesh.name = "Mahesh Parashar";
            $scope.Mahesh.rollno  = 1;

            $scope.Piyush = {};
            $scope.Piyush.name = "Piyush Parashar";
            $scope.Piyush.rollno  = 2;
         });
      </script>
   </body>
</html>
```

Project Examples: @@@@@@@@Case Study Based Learning@@@@@@@@@@@@@

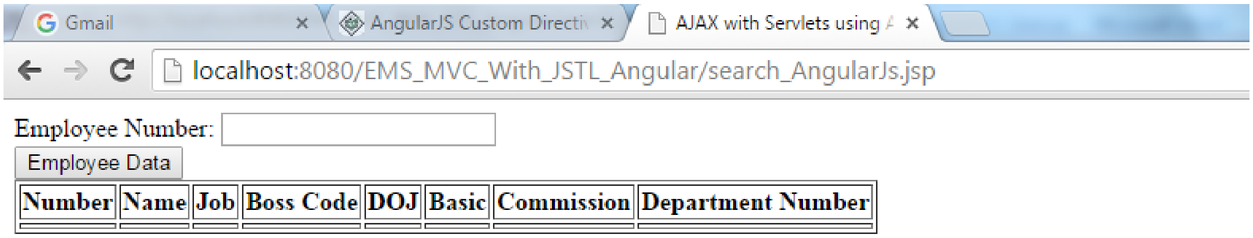
SearchEmployee Module:

```
<!DOCTYPE html>
<html><head><meta charset="ISO-8859-1">
<title>AJAX with Servlets using AngularJS</title>
<script type="text/javascript" src="js/angular.min.js"></script>
<script>
     var app = angular.module('myApp', []);
     function MyController($scope, $http) {
          $scope.getDataFromServer = function() {
               $http({
                    method : 'POST',
                    url : 'SearchController1',
                    params: {employeeNumber: $scope.employeeNumber},
               }).success(function(data, status, headers, config) {
                    $scope.employeeVO = data;
               }).error(function(data, status, headers, config) {
          // called asynchronously if an error occurs
               // or server returns response with an error status.
               });
          };
     };
</script></head>
<body><div data-ng-app="myApp">
          <div data-ng-controller="MyController">
          <div><label>Employee Number:</label>
          <input data-ng-model="employeeNumber" type="text">
</div><button data-ng-click="getDataFromServer()">
Employee Data</button><table border="1">
<tr><th>Number</th><th>Name</th><th>Job</th><th>Boss Code</th>
<th>DOJ</th><th>Basic</th><th>Commission</th><th>Dept Number</th>
               </tr><tr><td>{{employeeVO.empNumber}}</td>
               <td>{{employeeVO.empName}}</td>
               <td>{{employeeVO.empJob}}</td>
               <td>{{employeeVO.empBossCode}}</td>
               <td>{{employeeVO.empDoj}}</td>
               <td>{{employeeVO.empSalary}}</td>
               <td>{{employeeVO.empComm}}</td>
               <td>{{employeeVO.empDeptNumber}}</td>
               </tr>
          </table>
     </div>
     </div>
</body></html>
```

SearchController1.java: Servlet Code

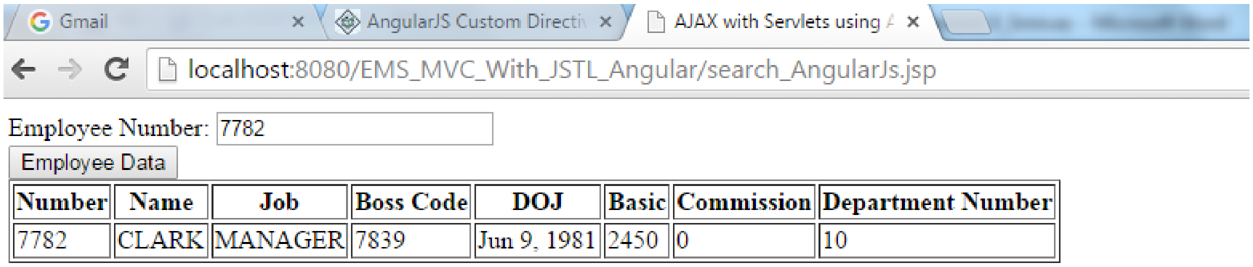
```java
package com.ems.controller;
import java.io.IOException;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.apache.log4j.Logger;
import com.ems.bo.SearchBO;
import com.ems.constants.ErrorConstants;
import com.ems.dao.EmployeeDAO;
import com.ems.exceptions.EMSBusinessException;
import com.ems.exceptions.EMSException;
import com.ems.vo.EmployeeVO;
import com.google.gson.Gson;

/**
 * Servlet implementation class SearchController1
 */
public class SearchController1 extends HttpServlet {
    private static final long serialVersionUID = 1L;

    private static final Logger LOG = 
Logger.getLogger(EmployeeDAO.class);
    private final SearchBO searchBO;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public SearchController1() {

        LOG.debug("From Search1 Controller Cons");
        searchBO = new SearchBO();
    }
    /**
     * @see HttpServlet#doPost(HttpServletRequest request,
HttpServletResponse
     *      response)
     */
    @Override
    protected void doPost(HttpServletRequest request,
            HttpServletResponse response) throws ServletException,
IOException {
        LOG.debug("I AM From SearchControllers doPost Method");
        try {
            int empNumber = 0;
            try {
                empNumber =

Integer.parseInt(request.getParameter("employeeNumber"));
                //empNumber = 7839;
            } catch (NumberFormatException e) {
                throw new EMSBusinessException(
                        ErrorConstants.DATA_FORMAT_EXCEPTION);
            }
    EmployeeVO employeeVO = searchBO.searchEmployee(empNumber);
            String json = new Gson().toJson(employeeVO);
            response.setContentType("application/json");
            System.out.println(json);
            response.getWriter().write(json);

        } catch (EMSException e) {
            String message = e.getMessage();
            LOG.error("Exception Caught In Controller", e);
            request.setAttribute("message", message);
            RequestDispatcher dispatcher = request
                    .getRequestDispatcher("./Error.jsp");
            dispatcher.forward(request, response);
        } catch (EMSBusinessException e) {
            LOG.error("Exception Caught In Controller", e);
            String message = e.getMessage();
            request.setAttribute("message", message);
            RequestDispatcher dispatcher = request
                    .getRequestDispatcher("./search.jsp");
            dispatcher.forward(request, response);
        }
    }
}
```

Output:


Gmail
AngularJS Custom Directi
AJAX with Servlets using A
localhost:8080/EMS_MVC_With_JSTL_Angular/search_AngularJs.jsp
Employee Data

Employee Number:

Number	Name	Job	Boss Code	DOJ	Basic	Commission	Department Number


Gmail
AngularJS Custom Directi
AJAX with Servlets using A
localhost:8080/EMS_MVC_With_JSTL_Angular/search_AngularJs.jsp
Employee Data

Employee Number: 7782

Number	Name	Job	Boss Code	DOJ	Basic	Commission	Department Number
7782	CLARK	MANAGER	7839	Jun 9, 1981	2450	0	10

DisplayEmployees Module:

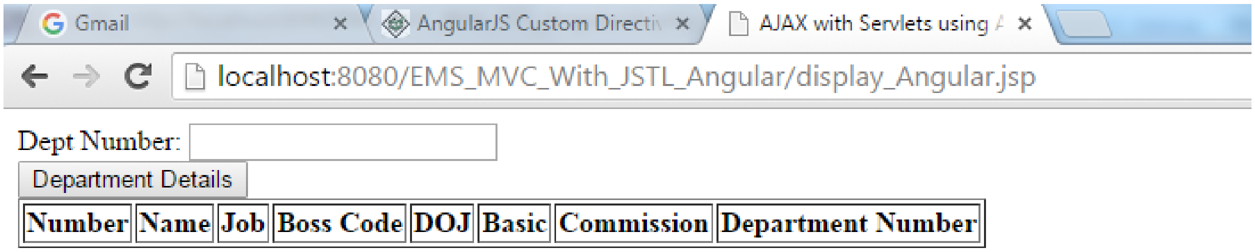
```
<!DOCTYPE html>
<html><head>
<meta charset="ISO-8859-1">
<title>AJAX with Servlets using AngularJS</title>
<script type="text/javascript" src="js/angular.min.js"></script>
<script>
    var app = angular.module('myApp', []);
    function MyController($scope, $http) {
        $scope.getDataFromServer = function() {
            $http({
                method : 'POST',
                url : 'DisplayController1',
                params: {deptNumber: $scope.deptNumber},
            }).success(function(data, status, headers, config) {
                $scope.empList=data;
                $scope.errorVO.errorMsg="";
                $scope.errorVO.statusCode="";
            }).error(function(data, status, headers, config) {
                $scope.errorVO=data;
                $scope.empList="";
                // called asynchronously if an error occurs
            // or server returns response with an error status.
            });
        };
    };
</script></head>
```

```
<body>
    <div data-ng-app="myApp">
        <div data-ng-controller="MyController">
            <div>
    <label>Dept Number:</label> <input data-ng-model="deptNumber"
                    type="text">
            </div>
            <button data-ng-click="getDataFromServer()">Department
                Details</button>
            <table>
<tr><td><font color="red">{{errorVO.errorMsg}}</font></td><td>
        <font color="red">{{errorVO.statusCode}}</font></td>
</tr></table>
            <table border="1">
                <tr><th>Number</th>
                    <th>Name</th>
                    <th>Job</th>
                    <th>Boss Code</th>
                    <th>DOJ</th>
                    <th>Basic</th>
                    <th>Commission</th>
                    <th>Department Number</th>
                </tr>
                <tr ng-repeat="employeeVO in empList">
                    <td>{{employeeVO.empNumber}}</td>
                    <td>{{employeeVO.empName}}</td>
                    <td>{{employeeVO.empJob}}</td>
                    <td>{{employeeVO.empBossCode}}</td>
                    <td>{{employeeVO.empDoj}}</td>
                    <td>{{employeeVO.empSalary}}</td>
                    <td>{{employeeVO.empComm}}</td>
                    <td>{{employeeVO.empDeptNumber}}</td>
                </tr>
            </table>
        </div>
    </div>
</body>
</html>
```

DisplayController1.java: ServletCode

```
package com.ems.controller;
import java.util.List;
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.apache.log4j.Logger;
import com.ems.bo.DisplayBO;
import com.ems.constants.ErrorConstants;
import com.ems.dao.EmployeeDAO;
import com.ems.exceptions.EMSBusinessException;
import com.ems.exceptions.EMSException;
import com.ems.vo.EmployeeVO;
import com.ems.vo.ErrorVO;
import com.google.gson.Gson;

/**
 * Servlet implementation class DisplayController1
 */
public class DisplayController1 extends HttpServlet {
	private static final long serialVersionUID = 1L;
	private static final Logger LOG = 
Logger.getLogger(EmployeeDAO.class);
	private final DisplayBO displayBO;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public DisplayController1() {
    	
        LOG.debug("From Display1Controller Cons");
        displayBO=new DisplayBO();
        
    }
    /**
     * @see HttpServlet#doPost(HttpServletRequest request, 
HttpServletResponse response)
     */
    protected void doPost(HttpServletRequest request, 
HttpServletResponse response) throws ServletException, IOException {
        LOG.debug("I AM From Display1Controller doPost Method");
        String json=null;
        response.setContentType("application/json");
        try {
            int empDeptNumber = 0;
            try {
empDeptNumber =Integer.parseInt(request.getParameter("deptNumber"));
                //empDeptNumber = 20;
            } catch (NumberFormatException e) {
                throw new EMSBusinessException(
                        ErrorConstants.DATA_FORMAT_EXCEPTION);
            }
List<EmployeeVO> 
empList=displayBO.displayEmployeesList(empDeptNumber);
             json=new Gson().toJson(empList);
            System.out.println(json);
            response.getWriter().write(json);

        } catch (EMSException e) {
            LOG.error("Exception Caught In Controller", e);
            ErrorVO vo=new ErrorVO();
            vo.setStatusCode(501);
            vo.setErrorMsg(e.getMessage());
            json=new Gson().toJson(vo);
            response.getWriter().write(json);
            System.out.println(json);

        } catch (EMSBusinessException e) {
            LOG.error("Exception Caught In Controller", e);
            ErrorVO vo=new ErrorVO();
            vo.setStatusCode(501);
            vo.setErrorMsg(e.getMessage());
            json=new Gson().toJson(vo);
            response.getWriter().write(json);
            System.out.println(json);
        }
    }
    }
```

Output:

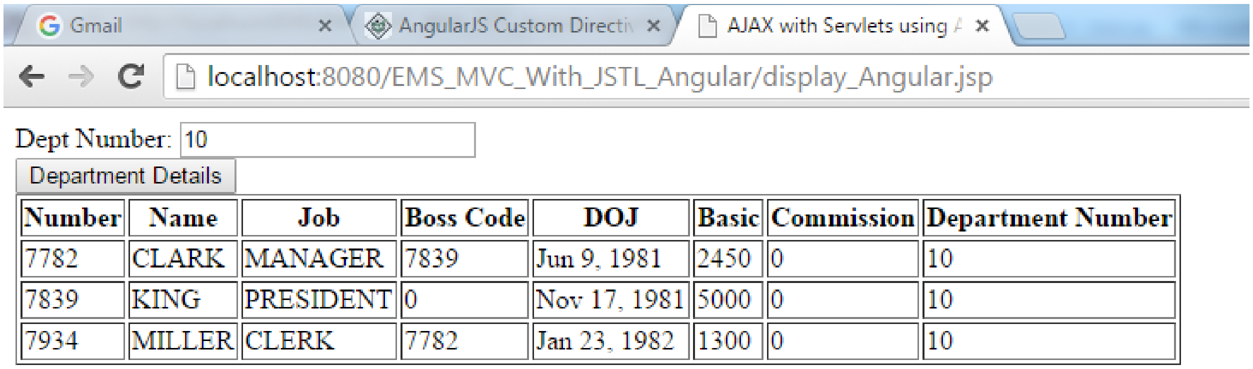


localhost:8080/EMS_MVC_With_JSTL_Angular/display_Angular.jsp

Dept Number:

Department Details

Number	Name	Job	Boss Code	DOJ	Basic	Commission	Department Number

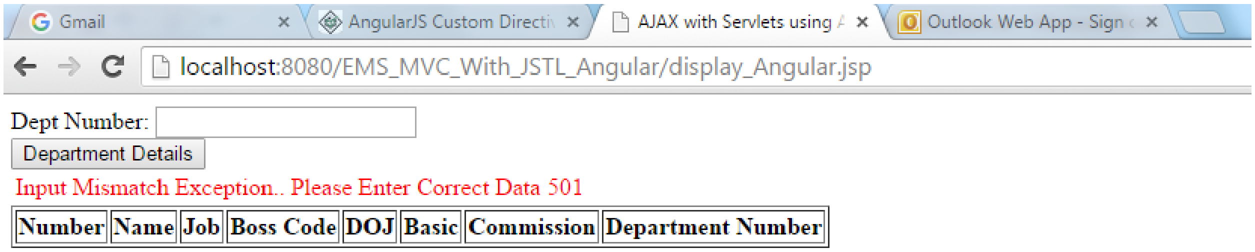


localhost:8080/EMS_MVC_With_JSTL_Angular/display_Angular.jsp

Dept Number: 10

Department Details

Number	Name	Job	Boss Code	DOJ	Basic	Commission	Department Number
7782	CLARK	MANAGER	7839	Jun 9, 1981	2450	0	10
7839	KING	PRESIDENT	0	Nov 17, 1981	5000	0	10
7934	MILLER	CLERK	7782	Jan 23, 1982	1300	0	10



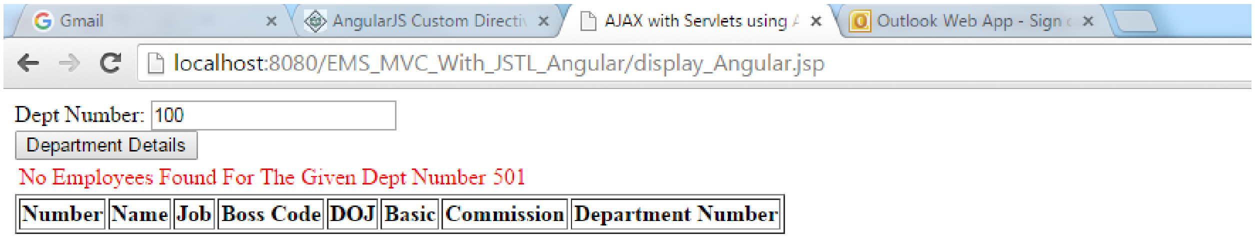
localhost:8080/EMS_MVC_With_JSTL_Angular/display_Angular.jsp

Dept Number:

Department Details

Input Mismatch Exception.. Please Enter Correct Data 501

Number	Name	Job	Boss Code	DOJ	Basic	Commission	Department Number



localhost:8080/EMS_MVC_With_JSTL_Angular/display_Angular.jsp

Dept Number: 100

Department Details

No Employees Found For The Given Dept Number 501

Number	Name	Job	Boss Code	DOJ	Basic	Commission	Department Number