**PART A**

(Part A: TO BE REFFERED BY STUDENTS)

**Experiment No. 09**

**A.1 AIM:**

Create an application to implement dependency injections and Services in Angular JS.

**A.2 Pre requisite:**

HTML, CSS, JavaScript

**A.3 Outcome:**

After successful completion of this experiment students will be able to:

1. Implement effective client-side implementation.
2. Solve the complex problem of development customized services in Angular JS.

**A.4 Theory:**

**Dependency Injection:** It is a software design in which components are given their dependencies instead of hard coding them within the component. It relieves a component from locating the dependency and makes dependencies configurable. It also 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 a simple JavaScript object, which is required to pass values to the controller during config phase (config phase is when AngularJS bootstraps itself).

//define a module

var mainApp = angular.module("mainApp", []);

//create a value object as "defaultInput" and pass it a data.

mainApp.value("defaultInput", 5);

...

//inject the value in the controller using its name "defaultInput"

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);

}

});

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

//define a module

var mainApp = angular.module("mainApp", []);

//create a factory "MathService" which provides a method multiply to return multiplication of two numbers

mainApp.factory('MathService', function() {

var factory = {};

factory.multiply = function(a, b) {

return a \* b

}

return factory;

});

//inject the factory "MathService" in a service to utilize the multiply method of factory.

mainApp.service('CalcService', function(MathService) {

this.square = function(a) {

return MathService.multiply(a,a);

}

});

...

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

//define a module

var mainApp = angular.module("mainApp", []);

...

//create a service which defines a method square to return square of a number.

mainApp.service('CalcService', function(MathService) {

this.square = function(a) {

return MathService.multiply(a,a);

}

});

//inject the service "CalcService" into the controller

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);

}

});

**Provider:** Provider is used by AngularJS internally to create services, factory, etc. during the config phase. The following script can be used to create MathService that we created earlier. Provider is a special factory method with get() method which is used to return the value/service/factory.

//define a module

var mainApp = angular.module("mainApp", []);

...

//create a service using provider which defines a method square to return square of a number.

mainApp.config(function($provide) {

$provide.provider('MathService', function() {

this.$get = function() {

var factory = {};

factory.multiply = function(a, b) {

return a \* b;

}

return factory;

};

});

});

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

mainApp.constant("configParam", "constant value");

**Service**

AngularJS supports the concept of Separation of Concerns using services architecture. Services are JavaScript functions, which are responsible to perform only specific tasks. This makes them individual entities which are maintainable and testable. The controllers and filters can call them on requirement basis. Services are normally injected using the 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 such as the $http is used to make ajax call to get the server data, the $route is used to define the routing information, and so on. The inbuilt services are always prefixed with $ symbol.

There are two ways to create a service −

* Factory
* Service

**Example:**

In AngularJS, a service is a function, or object, that is available for, and limited to, your AngularJS application. AngularJS has about 30 built-in services.

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

app.controller('myCtrl', function($scope, $timeout) {

$scope.myHeader = "Hello World!";

$timeout(function () {

$scope.myHeader = "How are you today?";

}, 2000);

});

**A.5 Procedure/Task:**

* Create an application to implement dependency injections method to build a customized service in Angular JS
* Create an application to implement inbuilt services in Angular JS. (Note: Services to implement are: $location, $timeout, $http, $interval)

Prepare the document. Save and close the file and name it as **EXP09\_Name of Student**

**PART B**

(PART B: TO BE COMPLETED BY STUDENTS)

(Students must submit the soft copy as per following segments within two hours of the practical. The soft copy must be uploaded on the Blackboard or emailed to the concerned lab in charge faculties at the end of the practical in case the there is no Black board access available)

|  |  |
| --- | --- |
| Roll No. : | Name: |
| Class : | Batch : |
| Date of Experiment : | Date/Time of Submission : |
| Grade : |  |

**B.1 Code:**

*(Paste your Code here)*

**B.2 Output**

*(Take screen shots of the output at run time and paste it here)*

**B.3 Conclusion:**

*(Students must write the conclusion as per the attainment of individual outcome listed above)*

**B.3 Observations and Learning:**

*(Students must write their observations and learnings as per the attainment of individual outcome listed above)*