

XII. THÀNH PHẦN VIEW TRONG ANGULARJS

- AngularJS hỗ trợ Single Page Application thông qua multiple view trên một trang đơn.
- Để làm được điều này, AngularJS cung cấp ng-view và ng-template directive và \$routeProvider service.

- Giới thiệu ng-view trong AngularJS:

- Thẻ ng-view đơn giản là tạo nơi giữ các màn hình view tương ứng có thể được đặt trong nó dựa vào cấu hình.
- Cách sử dụng

Định nghĩa thẻ div với ng-view trong module chính.

```
<div ng-app="ungdungAngularJS">
...
<div ng-view></div>

</div>
```

- Giới thiệu ng-template trong AngularJS:

- ng-template directive được sử dụng để tạo ra các HTML view sử dụng thẻ script. Nó chứa thuộc tính "id" được sử dụng bởi \$routeProvider để liên kết view và controller.
- Cách sử dụng

Định nghĩa một khối script với kiểu như ng-template trong module chính.

```
<div ng-app="ungdungAngularJS">
...
<script type="text/ng-template" id="themSV.html">
  <h2> Add Student </h2>
  {{message}}
</script>
```

</div>

- **Giới thiệu \$routeProvider trong AngularJS:**

- Là dịch vụ chính trong việc tạo các cấu hình cho địa chỉ URL, liên kết chúng với trang HTML tương ứng hoặc ng-template và gắn controller với chúng.
- Cách sử dụng

Định nghĩa một khối script trong module chính và thiết lập cấu hình định tuyến.

```
var ứng dụngAngularJS = angular.module("ứng dụngAngularJS", ['ngRoute']);

ứng dụngAngularJS.config(['$routeProvider',
  function($routeProvider) {
    $routeProvider.
      when('/themSV', {
        templateUrl: 'themSV.html',
        controller: 'themSVController'
      }).
      when('/quansatSV', {
        templateUrl: 'quansatSV.html',
        controller: 'quansatSVController'
      }).
      otherwise({
        redirectTo: '/themSV'
      });
  });
});
```

- Dưới đây là những điểm quan trọng cần xem xét từ ví dụ trên.

- \$routeProvider định nghĩa là một hàm dưới config của ứng dụngAngularJS module sử dụng khóa là "\$routeProvider".

- \$routeProvider.when định nghĩa một địa chỉ URL "/themSV" được sử dụng để liên kết đến trang "themSV.html", trong đó themSV.html nên đặt chung thư mục đường dẫn với trang HTML. Nếu trang HTML không được định nghĩa, ng-template sẽ sử dụng id="themSV.html". Chúng ta sử dụng ng-template.
 - "otherwise" được sử dụng để thiết lập view mặc định.
 - "controller" được để thiết lập controller tương ứng với từng view.
- Ví dụ Dưới đây là ví dụ minh họa cho những directive mô tả bên trên.

thanhphanView.html

```
<html>
<head>
  <title>Vi du View trong Angular JS</title>
  <script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>
  <script          src="http://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular -
route.min.js"></script>
</head>
<body>
  <h2>Ung dung AngularJS</h2>
  <div ng-app="ungdungAngularJS">
    <p><a href="#themSV">Them sinh vien</a></p>
    <p><a href="#quansatSV">Quan sat sinh vien</a></p>
    <div ng-view></div>
    <script type="text/ng-template" id="themSV.html">
      <h2> Them sinh vien </h2>
      {{message}}
    </script>
    <script type="text/ng-template" id="quansatSV.html">
      <h2> Quan sat sinh vien </h2>
      {{message}}
    </script>
  </div>
</body>
</html>
```

```
</script>
```

```
</div>
```

```
<script>
```

```
var ungdungAngularJS = angular.module("ungdungAngularJS", ['ngRoute']);
```

```
ungdungAngularJS.config(['$routeProvider',
```

```
function($routeProvider) {
```

```
    $routeProvider.
```

```
        when('/themSV', {
```

```
            templateUrl: 'themSV.html',
```

```
            controller: 'themSVController'
```

```
        }).
```

```
        when('/quansatSV', {
```

```
            templateUrl: 'quansatSV.html',
```

```
            controller: 'quansatSVController'
```

```
        }).
```

```
        otherwise({
```

```
            redirectTo: '/themSV'
```

```
        });
```

```
    });
```

```
ungdungAngularJS.controller('themSVController', function($scope) {
```

```
    $scope.message = "Trang nay se duoc su dung de hien thi mot form de them sinh vien";
```

```
});
```

```
ungdungAngularJS.controller('quansatSVController', function($scope) {
```

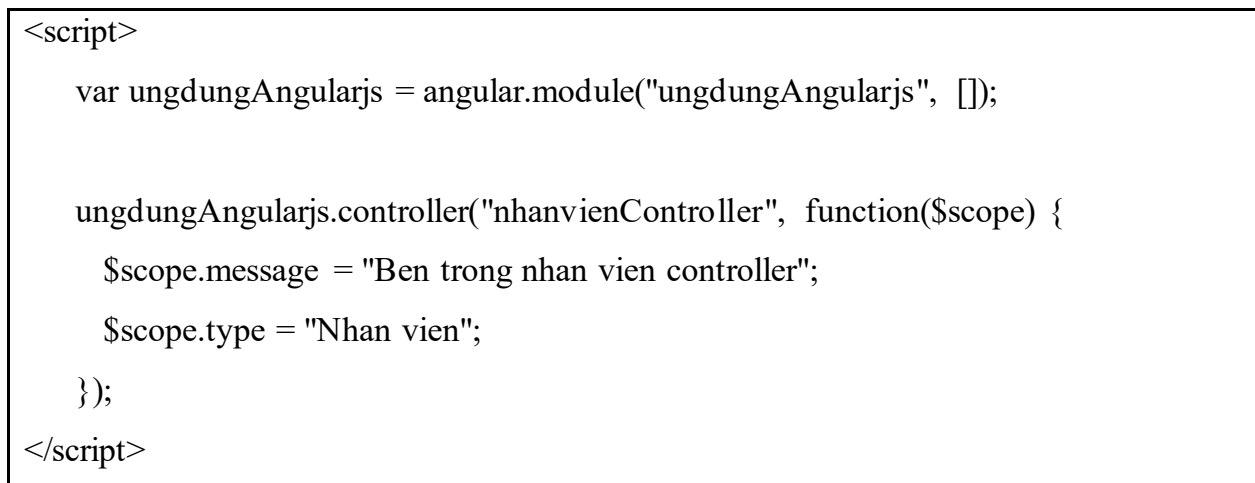
```
    $scope.message = "Trang nay se duoc su dung de quan sat tat ca sinh vien";
```

```
});
```



XIII. SCOPE TRONG ANGULARJS

- Scope là đối tượng JavaScript đặc biệt có vai trò liên kết controller và view.
- Scope chứa thông tin là các dữ liệu model. Trong controller, dữ liệu model có thể được truy cập qua đối tượng \$scope.



- Dưới đây là những điểm quan trọng của ví dụ trên.
 - o \$scope được truyền như là tham số đầu tiên của controller trong hàm khởi tạo của nó.
 - o \$scope.message và \$scope.type là các model được sử dụng trong trang HTML.

- Chúng ta thiết lập các giá trị cho model và tác động lên Module ứng dụng với controller và nhanvienController.
- Chúng ta có thể định nghĩa các hàm với \$scope.
- Tính kế thừa của Scope trong AngularJS
 - Scope là controller riêng biệt. Chúng ta định nghĩa nested controller (các controller lồng nhau) để các controller con sẽ kế thừa từ các controller cha..

```
<script>
    var ungdungAngularjs = angular.module("ungdungAngularjs", []);

    ungdungAngularjs.controller("nhanvienController", function($scope) {
        $scope.message = "Ben trong nhan vien controller";
        $scope.type = "Nhan vien";
    });

    ungdungAngularjs.controller("nhanvienITController", function($scope) {
        $scope.message = "Ben trong nhan vien IT controller";
    });
</script>
```

- Dưới đây là những điểm chính qua ví dụ trên.
 - Chúng ta tạo giá trị biến model cho nhanvienController.
 - Chúng ta ghi đề thông báo của controller con là nhanvienITController. Khi "message" được sử dụng trong các module của nhanvienITController, giá trị message ghi đề sẽ được sử dụng.
- Dưới đây là phần ví dụ minh họa cho phần hướng dẫn bên trên.

viduScope.html


```
<html>
<head>
  <title>Vi du Scope trong AngularJS</title>
</head>
<body>
  <h2>Ung dung AngularJS</h2>
  <div ng-app="ungdungAngularjs" ng-controller="nhanvienController">
    <p>{{message}} <br/> {{type}} </p>
    <div ng-controller="nhanvienITController">
      <p>{{message}} <br/> {{type}} </p>
    </div>
    <div ng-controller="nhanvienBHController">
      <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 ungdungAngularjs = angular.module("ungdungAngularjs", []);

    ungdungAngularjs.controller("nhanvienController", function($scope) {
      $scope.message = "Ben trong nhan vien controller";
      $scope.type = "Nhan vien";
    });

    ungdungAngularjs.controller("nhanvienITController", function($scope) {
      $scope.message = "Ben trong nhan vien IT controller";
    });

    ungdungAngularjs.controller("nhanvienBHController", function($scope) {
```

```
$scope.message = "Ben trong nhan vien ban hang controller";  
$scope.type = "Nhan vien BH";  
});  
  
</script>  
</body>  
</html>
```



Ung dung AngularJS

Ben trong nhan vien controller
Nhan vien

Ben trong nhan vien IT controller
Nhan vien

Ben trong nhan vien ban hang controller
Nhan vien BH

XIV. CÁC SERVICE TRONG ANGULARJS

- AngularJS hỗ trợ các khái niệm "Seperation of Concerns - Chia để trị" sử dụng cấu trúc service. Service là các hàm JavaScript và có nhiệm vụ trên những task nhất định. Nó làm cho chúng thành những thực thể riêng rẽ dễ dàng trong việc bảo trì và kiểm thử. Controller, filter có thể gọi chúng một cách đơn giản. Service thường được inject sử dụng cơ chế dependency injection của AngularJS.
- AngularJS cung cấp rất nhiều những service được định nghĩa cho trước: \$http,\$scope,\$route,\$window,\$location... Mỗi một service có những nhiệm vụ nhất định.
- Ví dụ, \$http được sử dụng để tạo ra các ajax request lên server để lấy dữ liệu về. \$route được sử dụng để định nghĩa thông tin routing Những service mặc định của AngularJS bắt đầu bởi biểu tượng \$.
- **Có 2 cách để tạo một service trong AngularJS là:**

factory

service

- **Sử dụng phương thức factory trong AngularJS**

- Khi sử dụng factory method, đầu tiên chúng ta định nghĩa factory và gán method cho nó.

```
var ungdungAngularJS = angular.module("ungdungAngularJS", []);
    ungdungAngularJS.factory('toanhocService', function() {
        var factory = {};
        factory.phepnhan = function(a, b) {
            return a * b
        }
        return factory;
    });
```

- **Sử dụng phương thức service trong AngularJS:**

- Sử dụng service method, chúng ta sẽ định nghĩa service sau đó gán method với nó. Chúng ta cũng inject những service có sẵn cho nó.

```
ungdungAngularJS.service('tinhBPService', function(toanhocService){
    this.binhphuong = function(a) {
        return toanhocService.phepnhan(a,a);
    }
});
```

- Ví dụ Dưới đây là ví dụ minh họa cho chỉ dẫn bên trên.

testAngularJS.jsp

```

<html>
<head>
  <title>Vi du Service trong AngularJS</title>
  <script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>
</head>
<body>
  <h2>Ung dung AngularJS</h2>
  <div ng-app="ungdungAngularJS" ng-controller="tinhBPController">
    <p>Nhap mot so: <input type="number" ng-model="number" />
    <button ng-click="binhphuong()">X<sup>2</sup></button>
    <p>Ket qua: {{ketqua}}</p>
  </div>
  <script>
    var ungdungAngularJS = angular.module("ungdungAngularJS", []);
    ungdungAngularJS.factory('toanhocService', function() {
      var factory = {};
      factory.phepnhan = function(a, b) {
        return a * b
      }
      return factory;
    });

    ungdungAngularJS.service('tinhBPService', function(toanhocService){
      this.binhphuong = function(a) {
        return toanhocService.phepnhan(a,a);
      }
    });

    ungdungAngularJS.controller('tinhBPController', function($scope, tinhBPService) {

```

```
$scope.binhphuong = function() {  
    $scope.ketqua = tinhBPService.binhphuong($scope.number);  
}  
});  
</script>  
</body>  
</html>
```

XV. DEPENDENCY INJECTION TRONG ANGULARJS

- Dependency Injection là một mô hình thiết kế phần mềm mà trong đó các thành phần được đưa ra từ những phần phụ thuộc nó - dependencies thay cho việc hard coding chúng trong các thành phần. Điều này làm cho các thành phần phụ thuộc nhau trong phần cấu hình. Nó giúp việc làm có các thành phần có tính tái sử dụng cao, dễ bảo dưỡng và kiểm thử.
- AngularJS cung cấp kỹ thuật Dependency Injection, cho phép các thành phần lõi của AngularJS có thể được inject tới các thành phần phụ thuộc khác.

value

factory

service

provider

constant

- Đối tượng value trong AngularJS

- o là đối tượng JavaScript đơn giản và được sử dụng để thiết lập các giá trị tới controller trong các bước cấu hình.

```
// Định nghĩa một module
```

```
var ứng dụngAngularjs = angular.module("ứng dụngAngularjs", []);
```

```
//tao mot doi tuong value duoi dang "giatrimacdinh" va truyen du lieu cho no.

ungdungAngularjs.value("giatrimacdinh", 5);

...

// inject gia tri nay trong controller boi su dung ten cua no la "giatrimacdinh"

ungdungAngularjs.controller('tinhBPController', function($scope, tinhBPService,
giatrimacdinh) {
    $scope.number = giatrimacdinh;
    $scope.ketqua = tinhBPService.binhphuong($scope.number);

    $scope.binhphuong = function() {
        $scope.ketqua = tinhBPService.binhphuong($scope.number);
    }
});
```

- Hàm factory trong AngularJS
 - o factory là một hàm để sử dụng trả về giá trị. Nó tạo ra giá trị theo yêu cầu mỗi khi service hoặc controller yêu cầu. Ta thường dùng các hàm factory để tính và trả về giá trị.

```
// Định nghĩa một module

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

// Tạo một factory là "toanhocService" mà cung cấp một phương thức là phép nhân để trả về tích của hai số

ungdungAngularjs.factory('toanhocService', function() {
```

```

var factory = {};
factory.phepnhan = function(a, b) {
    return a * b
}
return factory;
});

//inject "toanhocService" trong mot service de loi dung phuong thuc phepnhan cua factory.

ungdungAngularjs.service('tinhBPService', function(toanhocService){
    this.binhphuong = function(a) {
        return toanhocService.phepnhan(a,a);
    }
});
...

```

- **Đối tượng service trong AngularJS**

- service là một đối tượng singleton javascript chứa tập các hàm cho các mục đích cụ thể. Service được định nghĩa sử dụng hàm service() và sau đó inject nó đến controller.

```

// Định nghĩa một module

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

// Tạo một service mà định nghĩa một phương thức binhphuong để trả về bình phương của một số.

```

```

ungdungAngularjs.service('tinhBPService', function(toanhocService){
    this.binhphuong = function(a) {
        return toanhocService.phepnhan(a,a);
    }
});

//inject "tinhBPService" vào trong controller

ungdungAngularjs.controller('tinhBPController', function($scope, tinhBPService,
giatrimacdinh) {
    $scope.number = giatrimacdinh;
    $scope.ketqua = tinhBPService.binhphuong($scope.number);

    $scope.binhphuong = function() {
        $scope.ketqua = tinhBPService.binhphuong($scope.number);
    }
});

```

- Hàm provider trong AngularJS

- provider được sử dụng bởi trong nội bộ AngularJS để tạo service, factory ... trong quá trình cài đặt (quá trình mà AngularJS khởi tạo chính nó). Dưới đây mô tả script có thể tạo toanhocService trong đó chúng ta tạo trước đó. Provider là một phương thức factory đặc biệt với phương thức get() trả về giá trị là value/service/factory.

```

// Định nghĩa một module
var ungdungAngularjs = angular.module("ungdungAngularjs", []);
...

```

```
// Tao mot service boi su dung provider ma dinh nghia phuong thuc binhphuong de tra ve binh
phuong cua mot so.
ungdungAngularjs.config(function($provide) {
    $provide.provider('toanhocService', function() {
        this.$get = function() {
            var factory = {};
            factory.phiepnhan = function(a, b) {
                return a * b;
            }
            return factory;
        };
    });
});
```

- **constant trong AngularJS**

- o constant được sử dụng để truyền các giá trị trong tại giai đoạn cấu hình.

```
ungdungAngularjs.constant("thamsoCauHinh", "gia tri hang");
```

- Dưới đây là ví dụ minh họa các khái niệm trên:

```
<html>
<head>
    <title>Vi du Dependency Injection trong AngularJS</title>
</head>
<body>
    <h2>Ung dung AngularJS</h2>
    <div ng-app="ungdungAngularjs" ng-controller="tinhBPController">
        <p>Nhap mot so: <input type="number" ng-model="number" />
```

```
<button ng-click="binhphuong()">X<sup>2</sup></button>
<p>Ket qua: {{ketqua}}</p>
</div>
<script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>
<script>
    var ungdungAngularjs = angular.module("ungdungAngularjs", []);

    ungdungAngularjs.config(function($provide) {
        $provide.provider('toanhocService', function() {
            this.$get = function() {
                var factory = {};
                factory.phepnhan = function(a, b) {
                    return a * b;
                }
                return factory;
            };
        });
    });

    ungdungAngularjs.value("giatrimacdinhh", 5);

    ungdungAngularjs.factory('toanhocService', function() {
        var factory = {};
        factory.phepnhan = function(a, b) {
            return a * b;
        }
        return factory;
    });
```



```
    undungAngularjs.service('tinhBPService', function(toanhocService){
        this.binhphuong = function(a) {
            return toanhocService.phepnhan(a,a);
        }
    });

    undungAngularjs.controller('tinhBPController', function($scope, tinhBPService,
giatrimacdinhh) {
        $scope.number = giatrimacdinhh;
        $scope.ketqua = tinhBPService.binhphuong($scope.number);

        $scope.binhphuong = function() {
            $scope.ketqua = tinhBPService.binhphuong($scope.number);
        }
    });
</script>
</body>
</html>
```