AngularJS Scopes

Topics : AngularJS
Written on January 09, 2024

In AngularJS, scopes are objects that refer to the model and act as a glue between the controller and the view. They are an essential part of the two-way data binding mechanism, allowing changes in the model to be reflected in the view and vice versa. Here's a guide on how scopes work in AngularJS:

Basic Scope Usage:

In your AngularJS controller, you can attach properties and methods to the scope object, making them accessible in the associated view.

```
<!DOCTYPE html>
<html lang="en" ng-app="myApp">
<head>
<meta charset="UTF-8">
<title>AngularJS Scopes</title>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
</head>
<body ng-controller="MyController">
<p>{{ message }}</p>
<button ng-click="changeMessage()">Change Message</button>

<script>
angular.module('myApp', []).controller('MyController', function($scope) {
$scope.message = 'Hello, Angular!';

$scope.changeMessage = function() {
$scope.message = 'New Message!';
};
});
</script>
</body>
</html>
```

In this example:

- The `ng-controller` directive associates the `MyController` controller with the HTML body.
- The `{{ message }}` expression in the HTML binds to the `message` property of the scope.
- The `ng-click` directive triggers the `changeMessage` function when the button is clicked.

Nested Scopes:

AngularJS supports nested scopes, where a child scope inherits properties from its parent scope. This inheritance facilitates the hierarchical structure of controllers and views.

```
<!DOCTYPE html>
<html lang="en" ng-app="myApp">
<head>
<meta charset="UTF-8">
<title>AngularJS Nested Scopes</title>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
</head>
<body ng-controller="ParentController">
<p>{{ parentMessage }}</p>
<div ng-controller="ChildController">
<p>{{ childMessage }}</p>
</div>

<script>
angular.module('myApp', []).controller('ParentController', function($scope) {
$scope.parentMessage = 'Parent Message';
});

angular.module('myApp').controller('ChildController', function($scope) {
$scope.childMessage = 'Child Message';
});
</script>
</body>
</html>
```

In this example, the `ChildController` is nested inside the `ParentController`. The child scope inherits the properties of the parent scope.

Scope Inheritance:

When a child scope is created, it prototypically inherits properties from its parent scope. Any changes made in the child scope do not affect the parent scope directly, but changes in the parent scope are reflected in the child scope.

```
<!DOCTYPE html>
<html lang="en" ng-app="myApp">
<head>
<meta charset="UTF-8">
<title>AngularJS Scope Inheritance</title>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
</head>
<body ng-controller="ParentController">
<p>{{ sharedMessage }}</p>
<div ng-controller="ChildController">
<p>{{ sharedMessage }}</p>
</div>
```

```
<script>
angular.module('myApp', []).controller('ParentController', function($scope) {
$scope.sharedMessage = 'Shared Message';
});

angular.module('myApp').controller('ChildController', function($scope) {
// Child scope inherits the sharedMessage property
});
</script>
</body>
</html>
```

`$rootScope`:

AngularJS also provides a `$rootScope`, which is the top-level scope accessible to all controllers in an AngularJS application. It's generally recommended to avoid using `$rootScope` for application-specific data and prefer passing data through controllers or services.

Scope Lifecycle:

Scopes have a lifecycle that includes events such as `$destroy`, which is triggered when a scope is destroyed. You can use this event to perform cleanup tasks.

```
$scope.$on('$destroy', function() {
// Cleanup tasks when the scope is destroyed
});
```