ARYATECHNO

# AngularJS Controllers

**Topics :** AngularJS
**Written on** January 09, 2024

In AngularJS, controllers play a crucial role in the Model-View-Controller (MVC) architecture. Controllers are responsible for managing the application's logic, handling user input, and interacting with the model to update the view. They act as a bridge between the model (data) and the view (user interface).

1. **Controller Definition:**
   - **Description:** Controllers are defined using the `controller` function provided by AngularJS. They are attached to specific portions of the HTML using the `ng-controller` directive.
   - **Example:**

     ```
     angular.module('myApp').controller('MyController', function($scope) {
     // Controller logic goes here
     });
     ```

2. **Scope:**
   - **Description:** Controllers interact with the view through a special object called the scope. The scope is an execution context for expressions, and changes made to the scope are automatically reflected in the view.
   - **Example:**

     ```
     angular.module('myApp').controller('MyController', function($scope) {
     $scope.message = 'Hello, Angular!';
     });
     ```

     In this example, the `message` variable on the scope can be accessed in the associated HTML template.

3. **Dependency Injection:**
   - **Description:** Controllers can receive dependencies through dependency injection. Common dependencies include services, which handle business logic and data manipulation.
   - **Example:**

```
angular.module('myApp').controller('MyController', function($scope, myService) {
$scope.data = myService.getData();
});
```

4. **Handling User Input:**
   - **Description:** Controllers handle user interactions such as button clicks, form submissions, and other events. They define functions on the scope to be called in response to user actions.
   - **Example:**

     ```
     angular.module('myApp').controller('MyController', function($scope) {
     $scope.handleClick = function() {
     // Logic to handle button click
     };
     });
     ```

     In the associated HTML template:

     ```
     <button ng-click="handleClick()">Click me</button>
     ```

5. **Controller As Syntax:**
   - **Description:** AngularJS supports the "controller as" syntax, allowing you to assign a controller instance to a variable in the view. This syntax is useful for avoiding scope-related issues.
   - **Example:**

     ```
     angular.module('myApp').controller('MyController', function() {
     this.message = 'Hello, Angular!';
     });
     ```

     In the associated HTML template:

     ```
     <div ng-controller="MyController as ctrl">
     {{ ctrl.message }}
     </div>
     ```

6. **Controller Lifecycle:**
   - **Description:** Controllers have a lifecycle, and AngularJS provides hooks such as `controller` and `link` for performing actions during different stages of the controller's life.
   - **Example:**

     ```
     angular.module('myApp').controller('MyController', function($scope) {
     // Initialization logic

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

```
        });
```