AngularJS Tables

Topics : AngularJS
Written on January 09, 2024

In AngularJS, tables are commonly used to display tabular data, and AngularJS provides tools and directives to make working with tables more dynamic and efficient. Here, I'll outline how you can use AngularJS to create tables and enhance their functionality.

Basic Table Structure:

```
<!DOCTYPE html>
<html lang="en" ng-app="myApp">
<head>
<meta charset="UTF-8">
<title>AngularJS Tables</title>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.8.2/angular.min.js"></script>
</head>
<body ng-controller="TableController">
<table>
<thead>
<tr>
<th>Name</th>
<th>Email</th>
</tr>
</thead>
<tbody>
<tr ng-repeat="user in users">
<td>{{ user.name }}</td>
<td>{{ user.email }}</td>
</tr>
</tbody>
</table>

<script>
angular.module('myApp', []).controller('TableController', function($scope) {
$scope.users = [
{ name: 'John Doe', email: 'john@example.com' },
{ name: 'Jane Doe', email: 'jane@example.com' },
// Add more user data as needed
];
});
</script>
</body>
```

</html>

In this example:

- The `ng-repeat` directive is used to iterate over the `users` array, generating a row for each user in the table body.
- Data binding (`{{ user.name }}` and `{{ user.email }}`) is used to display user information.

Sorting and Filtering:

AngularJS provides built-in directives for sorting and filtering data in tables. Here's an example using the `orderBy` and `filter` filters:

<th>
Name

</th>

In this example, clicking on the "Name" header toggles the sorting order, and an arrow icon indicates the sorting direction.

Pagination:

For large datasets, pagination is often used to display a limited number of rows per page. AngularJS doesn't provide built-in pagination, but you can implement it using a custom filter and controller logic.

<tr ng-repeat="user in filteredUsers = (users | startFrom:(currentPage-1)*pageSize | limitTo:pageSize)">
<!-- Display user data -->
</tr>

<!-- Add pagination controls -->

<li ng-repeat="page in getPages()" ng-class="{ active: page === currentPage }">
{{ page }}

In this example:

- The `startFrom` custom filter is used to slice the array and display only the relevant rows based on the current page.
- Pagination controls, such as "Next" and "Previous" buttons, can be implemented using a controller with logic to update the current page.

Editable Table:

If you want to make your table editable, you can use the `ng-model` directive for two-way data binding and `ng-click` to trigger editing.

```
<td ng-click="editUser(user)">
<span ng-show="!user.editing">{{ user.name }}</span>
<input ng-show="user.editing" ng-model="user.name">
</td>
```

In this example, clicking on a cell switches between displaying the data and an input field for editing.